# BLAISE PASCAL MAGAZINE 98

Multi platform /Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux


# CONTENT

## ARTICLES



## ADVERTISERS




Niklaus Wirth

Pascal is an imperative and procedural programming language, which Niklaus Wirth designed (left below) in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator: Blaise Pascal (see top right).

Publisher: PRO PASCAL FOUNDATION in collaboration © Stichting Ondersteuning Programmeertaal Pascal - Netherlands

**Stephen Ball**
http://delphiaball.co.uk
@DelphiABall

**Dmitry Boyarintsev**
dmitry.living @ gmail.com

**Michaël Van Canneyt,**
michael @ freepascal.org

**Marco Cantù**
www.marcocantu.com
marco.cantu @ gmail.com

**David Dirkse**
www.davdata.nl
E-mail: David @ davdata.nl

**Benno Evers**
b.evers @ everscustomtechnology.nl

**Bruno Fierens**
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

**Holger Flick**
holger @ flixments.com

**Primož Gabrijelčič**
primoz @ gabrijelcic.org

**Mattias Gärtner**
nc-gaertnma@netcologne.de

**Max Kleiner**
www.softwareschule.ch
max @ kleiner.com

**John Kuiper**
john_kuiper @ kpnmail.nl

**Wagner R. Landgraf**
wagner @ tmssoftware.com

**Vsevolod Leonov**
vsevolod.leonov@mail.ru

**Andrea Magni** www.andreamagni.eu
andrea.magni @ gmail.com
www.andreamagni.eu/wp

**Paul Nauta PLM Solution** Architect
CyberNautics
paul.nauta @ cybernautics.nl

**Kim Madsen**
www.component4developers.com

**Boian Mitov**
mitov @ mitov.com

**Jeremy North**
jeremy.north @ gmail.com

**Detlef Overbeek - Editor in Chief**
www.blaisepascal.eu
editor @ blaisepascal.eu

**Howard Page Clark**
hdpc @ talktalk.net

**Heiko Rompel**
info @ rompelsoft.de

**Wim Van Ingen Schenau -Editor**
wisone @ xs4all.nl

**Peter van der Sman**
sman @ prisman.nl

**Rik Smit**
rik @ blaisepascal.eu

**Bob Swart**
www.eBob42.com
Bob @ eBob42.com

**B.J. Rao**
contact @ intricad.com

**Daniele Teti**
www.danieleteti.it
d.teti @ bittime.it

**Anton Vogelaar**
ajv @ vogelaar-electronics.com

**Danny Wind**
dwind @ delphicompany.nl

**Siegfried Zuhr**
siegfried @ zuhr.nl



Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to **editor@blaisepascal.eu**

**Editors**
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

**Correctors**
Howard Page-Clark, Peter Bijlsma





**Subscriptions** ( 2019 prices )

| | Internat. excl. VAT | Internat. incl. 9% VAT | Shipment |
|---|---|---|---|
| **Printed Issue ±60 pages** | € 155,96 | € 250 | € 80,00 |
| **Electronic Download Issue 60 pages** | € 64,20 | € 70 | ——— |
| **Printed Issue inside Holland (Netherlands) 60 pages** | ——— | € 250,00 | € 70,00 |

Subscriptions can be taken out online at **www.blaisepascal.eu** or by written order, or by sending an email to **office@blaisepascal.eu**
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
**Subscriptions run 365 days.** Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
**ABN AMRO Bank Account no. 44 19 60 863** or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
**IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147** (Stichting Programmeertaal Pascal)
**Subscription department**
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 **office@blaisepascal.eu**

Member and donator of WIKIPEDIA
Member of the **Royal Dutch Library**
KB

# From your editor

This is the last issue before the winter season starts. The sunflowers are a good example for that.
In this issue 98 you will find the last part of Python for Delphi. Python is really a very good tool and can be used as scripting language as well as a complete program.
I had in mind to create a simple Password Generator, and then looked at the world behind it: fascinating, and fantastic. It's always interesting to go back in history to be able to understand why things are the way they are.

A friend presented a book, which I read with great interest: "The Innovators"
written by Walter Isaacson - Publishers: Simon and Schuster.
There is also a Dutch translation: De uitvinders; and a German Translation:
Die Vordenker der digitalen Revolution von Ada Lovelace bis Steve Jobs. For who has real interest in history and understanding computing and how it evolved. It's a fantastic book I devoured, 620 pages of reading details about the beginning of computers, and I found that the originator of computing is a women: Ada, - Lady Lovelace.

To come back to modern times; Delphi11 Alexandria has arrived. In this issue I try to explain quite a number of new things which come along with the new Windows11. I think it will take several years before the public will move over. I see some Windows 3 still being in use…
For Company's I think, there is very good chance to achieve better security, less vulnerability and less leakage. But it will be a costly process. Lots of Governmental Organisations and Company, Factory's etc. are still running on platforms which can be found in the Pleistocene.

We just updated to the newest engine because new machines have a much smaller footprint and use much less energy even though they are much more potent and produce therefore much less heath and are very quiet. However it costs time and money to move over.

Since many of us do have to make a lot of versions of our apps before they can be launched it would be wise to make use of a repository. So I asked Michael van Canneyt to write about it. Its not a simple thing to use and needs a lot of explaining. This second article will be followed by another 3rd one in the next issue 99/100, so the functionality will be completely handled in these three articles.
Its for universal use: an FNC Component.
It can be used with any project whether it is Delphi or Lazarus.

I will try to make the next issue an issue you wont forget: It will be filled with great news we are working on, starting at the extended RTTI for Free Pascal, - Delphi Compatible- , attributes, new component from TMS. Bruno and his team especially created this for a little app which I will present to you and lots more. It will be a festival of news.
We will have lots of nice items for our subscribers.

If possible I want to organize in the coming November a first event for this year. I plan it but …
I hope we will be able to do so, because we have to know if we are allowed to, by November…

I'll announce that asap.

Detlef

From our Technical advisor: Cartoons from Jerry King


*"Your Long Covid has caused your memory to fade? With all that's going on in the world lately, I wish mine would."*

# Advertisement

## maXbox Starter86_3

Try except end. — Max

Thanks to Python4Delphi we now can evaluate (for expressions) or exec (for statements) some Python code in our scripts. This version 4.7.5.80 July 2021 allows us with the help of a Python Dll and an environment with modules in site-packages execute Py-functions. But the most is only available in a 32-bit space as maXbox is still 32-bit, possible also with 64-bit Python means the call of the external shell (ExecuteShell) with installed Python versions to choose from. By the way also a Python4Lazarus is available.

Imagine you need a 512-bit hash and you don't have the available function. SHA256 or SHA512 is a secure hash algorithm which creates a fixed length one way string from any input data.
OK you start the Python-engine in your maXbox script and load the DLL.
Most of the time you don't need to install Python cause you find a DLL or subdirectory for example in the Wow64 subsystem or in mySQL and load it. WoW64 (Windows 32-bit on Windows 64-bit) is a subsystem of the Windows operating system capable of running 32-bit applications on 64-bit Windows. To get a Dll that fits your size and space you can check with

```
writeln('is x64 '+botostr(Isx64('C:\maXbox\EKON25\python37.dll')));
```

You do also have helper functions in the unit PythonEngine.pas
as global subroutines to test the environment:

- GetPythonEngine (Returns the global TPythonEngine)
- PythonOK (checks engine init)
- PythonToDelphi
- IsDelphiObject
- PyObjectDestructor
- FreeSubtypeInst
- PyType_HasFeature

```
function GetPythonEngine : TPythonEngine;
function PythonOK : Boolean;
function PythonToDelphi( obj : PPyObject ) : TPyObject;
function IsDelphiObject( obj : PPyObject ) : Boolean;
procedure PyObjectDestructor( pSelf : PPyObject); cdecl;
procedure FreeSubtypeInst(ob:PPyObject); cdecl;
procedure Register;
function PyType_HasFeature(AType : PPyTypeObject; AFlag : Integer): Boolean;
function SysVersionFromDLLName(const DLLFileName : string): string;
procedure PythonVersionFromDLLName( LibName: string; out MajorVersion, MinorVersion: integer);
```

FOR EXAMPLE THE PYTHONOK:

```
function PythonOK: Boolean;
begin
  Result:= Assigned( gPythonEngine ) and
       (gPythonEngine.Initialized or gPythonEngine.Finalizing);
end;
```

Or best you install the environment with:
**https://www.python.org/ftp/python/3.7.9/python-3.7.9.exe**
Python source code and installers are available for download for all versions!
I provide also just a Dll which we use most at:
**https://sourceforge.net/projects/maxbox/files/Examples/EKON/P4D/python37.dll/download**
Search for registered versions is possible with the function
**GetRegisteredPythonVersions: TPythonVersions;**
On 64-bit Windows the 32-bit **python27.dll** is really in
**C:\Windows\sysWOW64.**
But if you try opening the
**C:\Windows\system32\python27.dll**
in a 32-bit process, it'll open just fine.
If I'm not mistaken, WOW stands for Woodoo Of Windows.

```
//if PythonVersionFromPath(PYHOME, aPythonVersion, false) then begin

  if GetLatestRegisteredPythonVersion(aPythonVersion) then begin
    aPythonVersion.AssignTo(eng);
    writeln('APIVersion: '+itoa(TPythonEngine(eng).APIVersion));
    writeln('RegVersion: '+TPythonEngine(eng).RegVersion);
    writeln('RegVersion: '+TPythonEngine(eng).DLLName);
    //TPythonEngine(PythonEngine).LoadDLL;
  end;

>>>     APIVersion: 1013
        RegVersion: 3.6
        RegVersion: python36.dll
```

To make sure your install path of Python is the right one test it with OpenDll() passing the path and call explicitly OpenDll():

```
procedure TDynamicDll.LoadDll;
begin
   OpenDll( DllName );
end;
   eng.dllpath:= 'C:\maXbox\EKON25'
   eng.dllname:= 'python37.dll';
   eng.AutoLoad:= false;
 try
  eng.OpenDll('C:\maXbox\EKON25\python37.dll');
```

Let's follow the **Sha512** example as our topic and then you type the path, home and name of the Dll the given way:

```
with TPythonEngine.create(self) do begin
//Config Dll or Autoload
pythonhome:= PYHOME;
LoadDll;
writeln(pythonhome);
writeln(ExecModule);
pypara:= 'https://en.wikipedia.org/wiki/WoW64';
//pypara:= filetostring(exepath+'maXbox4.exe')
try
writeln(evalstr('__import__("math").sqrt(45)'));
writeln(evalstr('__import__("hashlib").sha256(b"'+
pypara+'").hexdigest().upper()'));
writeln(evalstr('__import__("hashlib").sha512(b"'+
pypara+'").hexdigest().upper()'));
except
eng.raiseError;
writeln(ExceptionToString(ExceptionType, ExceptionParam));
finally
free;
end;
end;
```

A better way would be to open the hashing file with **evalstr()** and open itself, so we open with with open!:

```
eng.Execstring('with open(r"'+exepath+'maXbox4.exe", "rb") as afile:'+
' fbuf = afile.read()');
println(eng.evalstr('__import__("hashlib").algorithms_available'));
println(eng.evalstr('__import__("hashlib").sha512('+
'fbuf).hexdigest().upper()'));
println(eng.evalstr('__import__("hashlib").sha1(fbuf).hexdigest().upper()'));

>>> 72342518C27207099612...
>>> 3E38A48072D4F828A4BE4A52320F092FE50AE9C3
```

So the second last line is the **Sha512** and the result is:
72342518C272070...
and so on. The important thing is the **evalstr()** function. The **eval()** allows us to execute arbitrary strings as **Python** code.
It accepts a source string and returns an object. But we can also import modules with the usefule inbuilt syntax 'import("hashlib")'.
**Note** that in **Python GUI by Python4maXbox,** to print the result, you just need to state the inbuilt print() or println() or writeln function, it's not enough just by return statement. The output is re-routed to memo2 console component in maXbox by print or write.

The eval() is not just limited to simple expression. We can execute functions, call methods, reference variables and so on. So we use this by using the __import__() built-in function. Note also that the computed hash is converted to a readable hexadecimal string by hexdigest().upper() and uppercase the hex-values in one line, amazing isn't it.
We step a bit further to exec a script in a script! If we call a file or an const Python command then we use ExecString(PYCMD); The script you can find at:
http://www.softwareschule.ch/examples/pydemo3.txt

The essence is a bit of script as a const:

```
const PYCMD = 'print("this is box")'+LB+
      'import sys'+LB+
      'f=open(r"1050pytest21_5powers.txt","w")'+LB+
      'f.write("Hello PyWorld_mX47580, \n")'+LB+
      'f.write("This data will be written on the file.")'+LB+
       'f.close()';
```

The LB = CR+LF; is important because we call it like a file or stream and exec() is cleaning (delete CR) and encoding the passing script afterwards, LF alone is also sufficient:

```
writeln('ExecSynCheck1 '+botostr(eng.CheckExecSyntax(PYCMD)));
eng.ExecString(PYCMD);
```

We also check the syntax before eval to prevent an exception like this: Exception:
`Access violation at address 6BA3BA66 in module 'python36.dll'.` or
`'python37_32.dll' Read of address 000000AD.`
Free the engine means destroying it calls Py_Finalize, which frees all memory allocated by the Python Dll. Or, if you're just using the Python API without the VCL wrappers like we do, you can probably just call Py_NewInterpreter on your TPythonInterface object to get a fresh execution environment without necessarily discarding everything done before!

By success of execute PYCMD a file (1050pytest21.txt) is written with some text so we executed line by line the PYCMD. When an application uses the SysUtils unit, most runtime errors are automatically converted into exceptions. Many errors that would otherwise terminate an application – such as insufficient memory, division by zero, and general protection faults – can be caught and handled by raiseError(). This is now the whole tester Procedure PYLaz_P4D_Demo3; but my key takeaway is that only use eval() with a trusted source!

```
Procedure PYLaz_P4D_Demo3;
//https://wiki.freepascal.org/Python4Delphi
var eng : TPythonEngine; out1: TPythonGUIInputOutput;
begin
 eng:= TPythonEngine.Create(Nil);
 out1:= TPythonGUIInputOutput.create(nil);
 out1.output:= pyMemo; //debugout.output;  //memo2;
 out1.RawOutput:= False;
 out1.UnicodeIO:= False;
 out1.maxlines:= 20;
 out1.displaystring('this string thing king');
 //eng.IO:= Out1;
 Out1.writeline('draw the line');
 try
  eng.LoadDll;
  eng.IO:= Out1;
  if eng.IsHandleValid then begin
   writeln('DLLhandle: '+botostr(eng.IsHandleValid));
   WriteLn('evens: '+ eng.EvalStringAsStr('[x**2 for x in range(15)]'));
   WriteLn('gauss: '+ eng.EvalStringAsStr('sum([x for x in range(101)])'));
   WriteLn('gauss2: '+ eng.EvalStr('sum([x % 2 for x in range(10100)])'));
   WriteLn('mathstr: '+ eng.EvalStr('"py " * 7'));
   WriteLn('builtins: '+ eng.EvalStr('dir(__builtins__)'));
   WriteLn('upperstr: '+ eng.EvalStr('"hello again".upper()'));
   WriteLn('workdir: '+ eng.EvalStr('__import__("os").getcwd()'));

   eng.ExecString('print("powers:",[x**2 for x in range(10)])');
   writeln('ExecSynCheck1 '+botostr(eng.CheckExecSyntax(PYCMD)));
   eng.ExecString(PYCMD);
   writeln('ExecSynCheck2 '+botostr(eng.CheckExecSyntax(myloadscript)));
   writeln('ExecSynCheck3 '+
      botostr(eng.CheckExecSyntax(filetostring(PYSCRIPT))));
   //eng.ExecString(filetostring(PYSCRIPT));
   writeln(eng.Run_CommandAsString('print("powers:",[x**2 for x in
                              range(10)])',eval_input));
   pymemo.update;
  end
   else writeln('invalid library handle! '+Getlasterrortext);
  writeln('PythonOK: '+botostr(PythonOK));
 except
   eng.raiseError;
   writeln('PyErr '+ExceptionToString(ExceptionType, ExceptionParam));
 finally
   eng.free;
 end;
 out1.free;
 //pyImport(PyModule);
end;
```

The procedure raiseError helps to find errors for example: Exception: : SRE __main__ module mismatch. Make sure you do not have any mismatch between Python interpreter version used (like 3.7) and the "re" python module (like 3.6.1). By the way the resolution of Dlls has changed in Python 3.8 for Windows. New in version 3.8: Previous versions of CPython would resolve Dlls using the default behavior for the current process. This led to inconsistencies, such as only sometimes searching PATH or the current working directory, and OS functions such as AddDllDirectory having no effect. Conclusion: The eval() method parses the expression passed to it and runs python expression(code) (but no statements) within the program. For you and for me 5 functions are crucial:

```
Function CheckEvalSyntax(const str:AnsiString):Boolean;
Function CheckExecSyntax(const str:AnsiString):Boolean;
 Procedure ExecString(const command:AnsiString););
  Procedure ExecString3(const command:AnsiString););//alias
   Procedure ExecStrings4(strings:TStrings););
    Function EvalStringAsStr(const command:AnsiString):string);//alias
     Function EvalStr(const command:AnsiString):string);
```

Also, consider a situation when you have imported os module in your python program like above WriteLn('workdir: '+eng.EvalStr('import("os").getcwd()'));. The os module provides portable way to use operating system functionalities like: read or write a file. But a single command can delete all files in your system! So eval expects an expression, import is a statement. That said, what you can trying is the following combination:

```
Println('exec as eval: '+eng.EvalStr('exec("import os as o")'));
 Println('exec: '+eng.EvalStr('o.getcwd()'));
 //>>> exec as eval: None
 //>>> exec: C:\maXbox\mX47464\maxbox4
 writeln('uuid: '+eng.evalstr('exec("import uuid") or str(uuid.uuid4())'));
 //>>> uuid: 3b2e10f9-0e31-4961-9246-00852fd508bd
```

You can use exec in eval instead if you intend to import the module or also ExecString(): it depends on the global or local namespace you set, means also the second line knows the import statement from first line:

```
eng.ExecString('import math');
 Println('evalexec: '+eng.EvalStr('dir(math)'));
```

When you use a float that doesn't have an exact binary float representation, the Decimal constructor cannot create an accurate decimal representation. For example:

And the same with an EvalExec:

```
import decimal
from decimal import Decimal

x = Decimal(0.1)
print(x)
```

```
pymemo.lines.add('Decimal: '+
        eng.EvalStr('__import__("decimal").Decimal(0.1)'));
>>> 0.1000000000000000055511151231257827021181583404541015625
```

At last a minimal configuration called "Pyonfly". The minimal configuration depends on your Python-installation and the UseLastKnownVersion property in TDynamicDll but once known it goes like this with raiseError to get the Python exceptions:

```
with TPythonEngine.Create(Nil) do begin
 pythonhome:= PYHOME;
 try
  loadDLL;
  Println('Decimal: '+
     EvalStr('__import__("decimal").Decimal(0.1)'));
 except
  raiseError;
 finally
  free;
 end;
end;
```

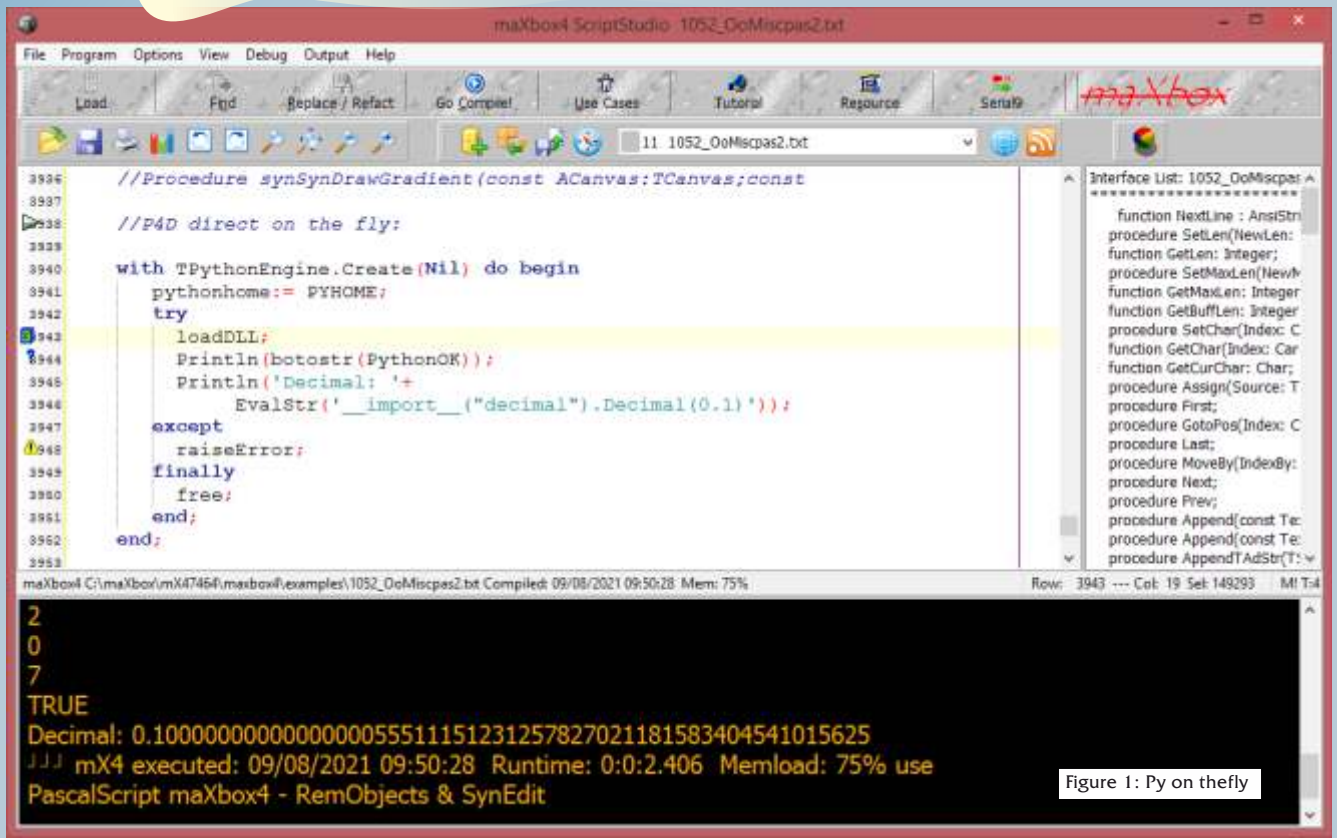



Figure 1: Py on thefly

The unit PythonEngine.pas is the main core-unit of the framework. Most of the Python/C API is presented as published/public member functions of the engine unit.

```
...
 Py_BuildValue              := Import('Py_BuildValue');
 Py_Initialize              := Import('Py_Initialize');
 PyRun_String               := Import('PyRun_String');
 PyRun_SimpleString         := Import('PyRun_SimpleString');
 PyDict_GetItemString       := Import('PyDict_GetItemString');
 PySys_SetArgv              := Import('PySys_SetArgv');
 Py_Exit                    := Import('Py_Exit');...
```

## TIPS:

NOTE: You will need to adjust the demos from github or sourceforge accordingly, to successfully load the Python distribution that you have installed on your computer so here's a small troubleshooter:

❶ Set a path first:

```
pydllpath= 'C:\Users\breitsch\AppData\Local\Programs\Python\Python37-32\python37.dll';
```

❷ Load it: `pythonengine.openDll(pydllpath);`

❸ Test it: `PrintLn('builtins: '+ pythonengine.EvalStr('dir(__builtins__)'));`

If you get the error:

```
Exception: :DLL load failed: %1 is not a valid Win32 application.
```

a solution is to set the `pythonhome` to 32bit:

```
PYHOME = 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
eng.pythonhome:= PYHOME;
```

Be sure that Pyhome and Pydll are of the same filespace when installing a package,e.g. install from within script, ex. numpy:

```
eng.ExecString('import subprocess');
eng.ExecString('subprocess.call(["pip", "install", "numpy"])');
eng.ExecString('import numpy');
```

Another complete 4 liner for environment testing:

```
eng.ExecString('import subprocess');
eng.ExecString('subprocess.call(["pip", "install", "langdetect"])')
eng.ExecString('from langdetect import detect');
println('detect: '+eng.EvalStr('detect("bonjour mes ordinateurs")'));
>>> detect: fr
```

IMPORTANT NOTE:

You should never pass untrusted source to the eval() directly. As it is quite easy for the malicious user to wreak havoc on your system. For example, the following code can be used to delete all the files from the system: `eval('os.system("RM -RF /")')`

## WIKI & EKON P4D TOPICS

- https://entwickler-konferenz.de/delphi-innovations-fundamentals/python4delphi/
- http://www.softwareschule.ch/examples/weatherbox.txt
- https://learndelphi.org/python-native-windows-gui-with-delphi-vcl/

## LEARN ABOUT PYTHON FOR DELPHI

- Tutorials
- Demos https://github.com/maxkleiner/python4delphi

Note: You will need to adjust the demos from github accordingly, to successfully load the Python distribution that you have installed on your computer.

Docs:
https://maxbox4.wordpress.com/blog/
http://www.softwareschule.ch/download/maxbox_starter86.pdf
http://www.softwareschule.ch/download/maxbox_starter86_1.pdf
http://www.softwareschule.ch/download/maxbox_starter86_2.pdf

**https://entwickler-konferenz.de/location-en/**


16 SEP Blog

### MACHINE LEARNING MIT CAI

This report visualizes the field of object recognition using computer vision techniques from machine learning. An image classifier from the CAI framework in Lazarus and Delphi, the so-called CIFAR-10 image classifier, is also used.

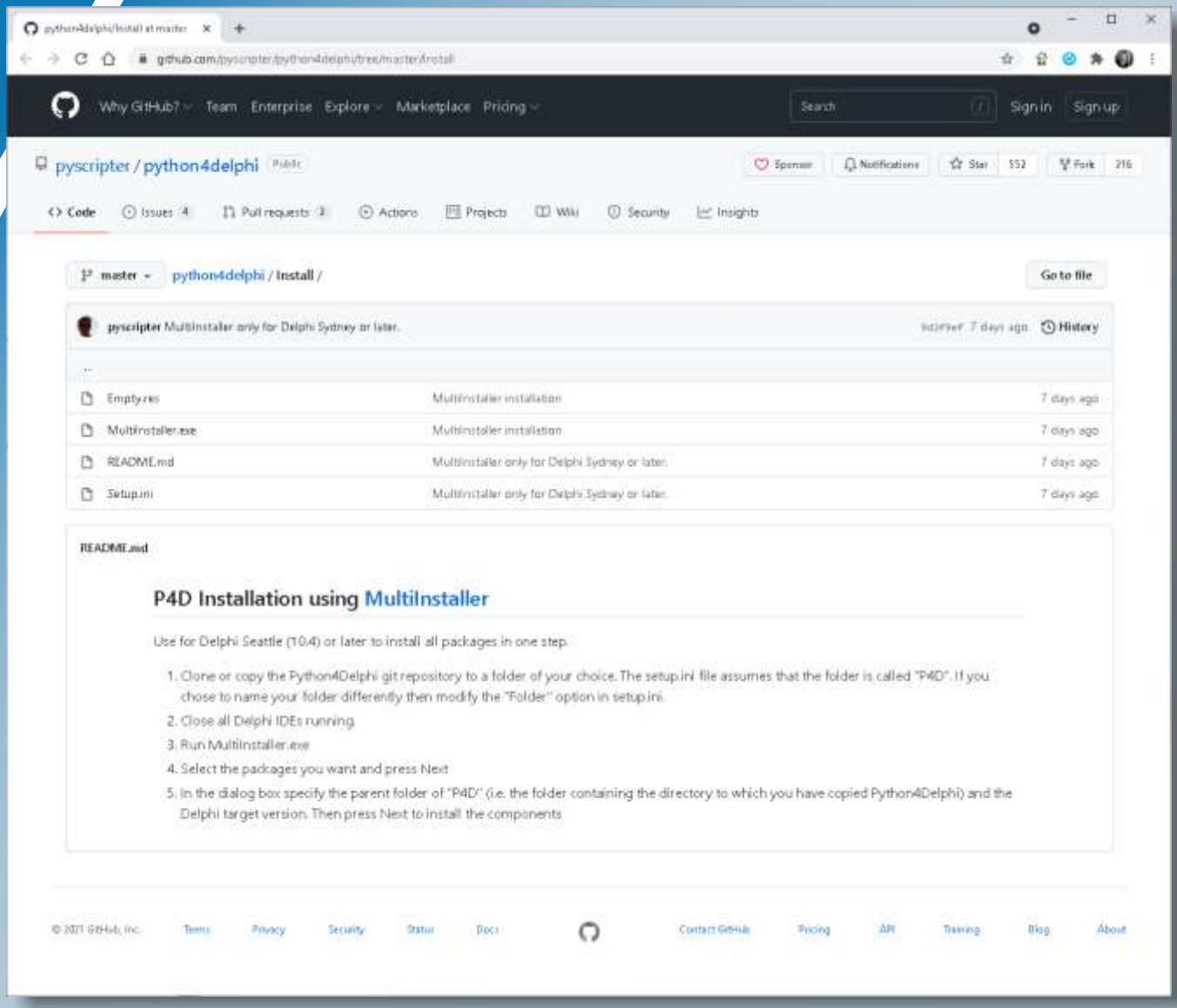
MEHR

18 SEP Blog, Interview

### "DELPHI DEVELOPMENT IS STILL GOING STRONG"

Marco Cantu talkes about the current status of Delphi, how it has evolved, and what's in store for this language in the future.

MEHR




**https://github.com/pyscripter/python4delphi/tree/master/Install**



WIKIPEDIA

**Guido van Rossum** (Dutch: born 31 January 1956) is a Dutch programmer best known as the creator of the Python programming language, for which he was the **"benevolent dictator for life"** (BDFL) until he stepped down from the position in July 2018. He remained a member of the Python Steering Council through 2019, and withdrew from nominations for the 2020 election.

Van Rossum was born and raised in the Netherlands, where he received a master's degree in mathematics and computer science from the University of Amsterdam in 1982. He has a brother, Just van Rossum, who is a type designer and programmer who designed the typeface used in the "Python Powered" logo.
Van Rossum lives in Belmont, California, with his wife, Kim Knapp, and their son. According to his home page and Dutch naming conventions, the "van" in his name is capitalized when he is referred to by surname alone, but not when using his first and last name together.

While working at the Centrum Wiskunde & Informatica (CWI), Van Rossum wrote and contributed a glob() routine to BSD Unix in 1986 and helped develop the ABC programming language. He once stated, "I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." He also created Grail, an early web browser written in Python, and engaged in discussions about the HTML standard.→


"Four Yorkshiremen sketch" at the 2014 Monty Python reunion. Written by Cleese, Chapman, Tim Brooke-Taylor and Marty Feldman, it was originally performed on their TV series At Last the 1948 Show in 1967. It parodies nostalgic conversations about humble beginnings or difficult childhoods.

## Wikipedia Continuation

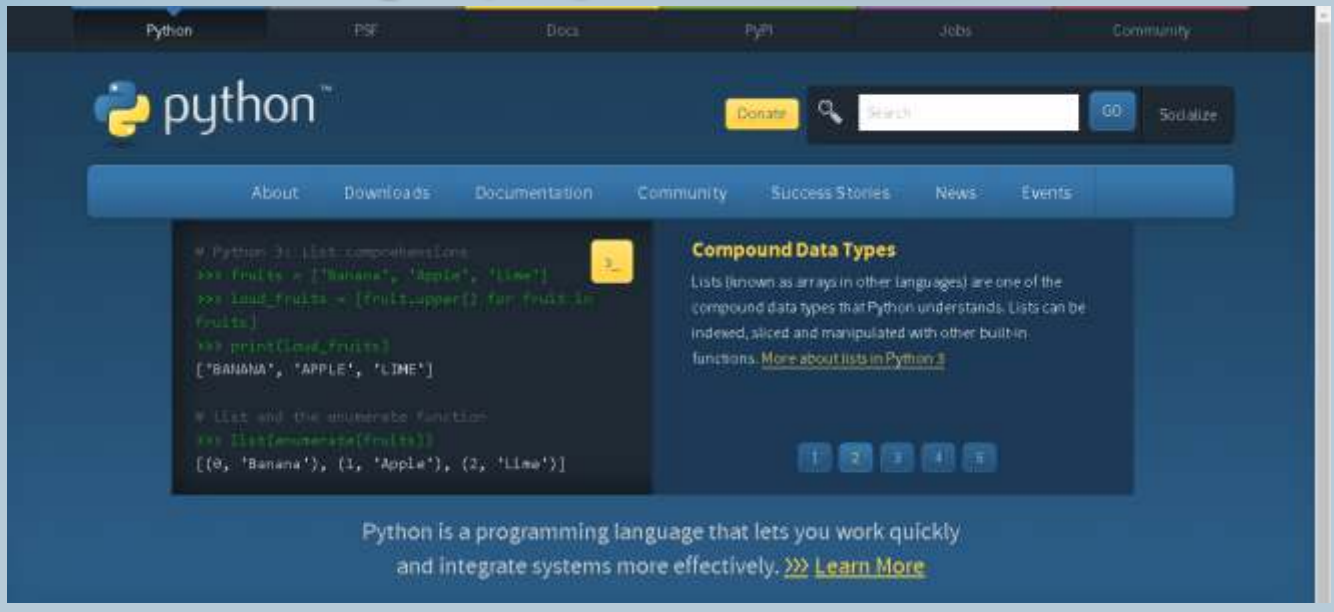
He has worked for various research institutes, including the Centrum Wiskunde & Informatica (CWI) in the Netherlands, the U.S. National Institute of Standards and Technology (NIST), and the Corporation for National Research Initiatives (CNRI).

From 2000 until 2003 he worked for the Zope corporation.
In 2003 Van Rossum left Zope for Elemental Security. While there he worked on a custom programming language for the organization.
From 2005 to December 2012, he worked at Google, where he spent half of his time developing the Python language.
In January 2013, he started working for Dropbox.
In October 2019, Van Rossum officially retired before coming out of retirement the following year to join Microsoft.

In December 1989, Van Rossum had been looking for a "'hobby' programming project that would keep him occupied during the week around Christmas" as his office was closed when he decided to write an interpreter for a "new scripting language, he had been thinking about lately:
a descendant of ABC that would appeal to Unix/C hackers".
He attributes choosing the name "Python" to "being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)".
He has explained that Python's predecessor, ABC, was inspired by SETL,
noting that ABC co-developer Lambert Meertens had "spent a year with the SETL group at NYU (New York University) before coming up with the final ABC design".

In July 2018, Van Rossum announced that he would be stepping down from the position of BDFL of the Python programming language.

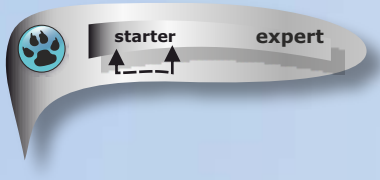
## And now for something completely different:



# HOW TO ALTER YOUR DEBUGGER FROM GNU TO FPC IN LAZARUS

BY MATTIAS GAERTNER






In the version of **Lazarus 22.0. RC1**(the latest available for this moment) has a little annoyance emerged:
the normal gnu debugger is very slow. So to solve this problem there is an easy way, but you should know how to use it: Before all you need to test if there is a package called **LazDebuggerFp** installed.
If not, install it, (see below: in the Menu go to **Package Package Graph**)

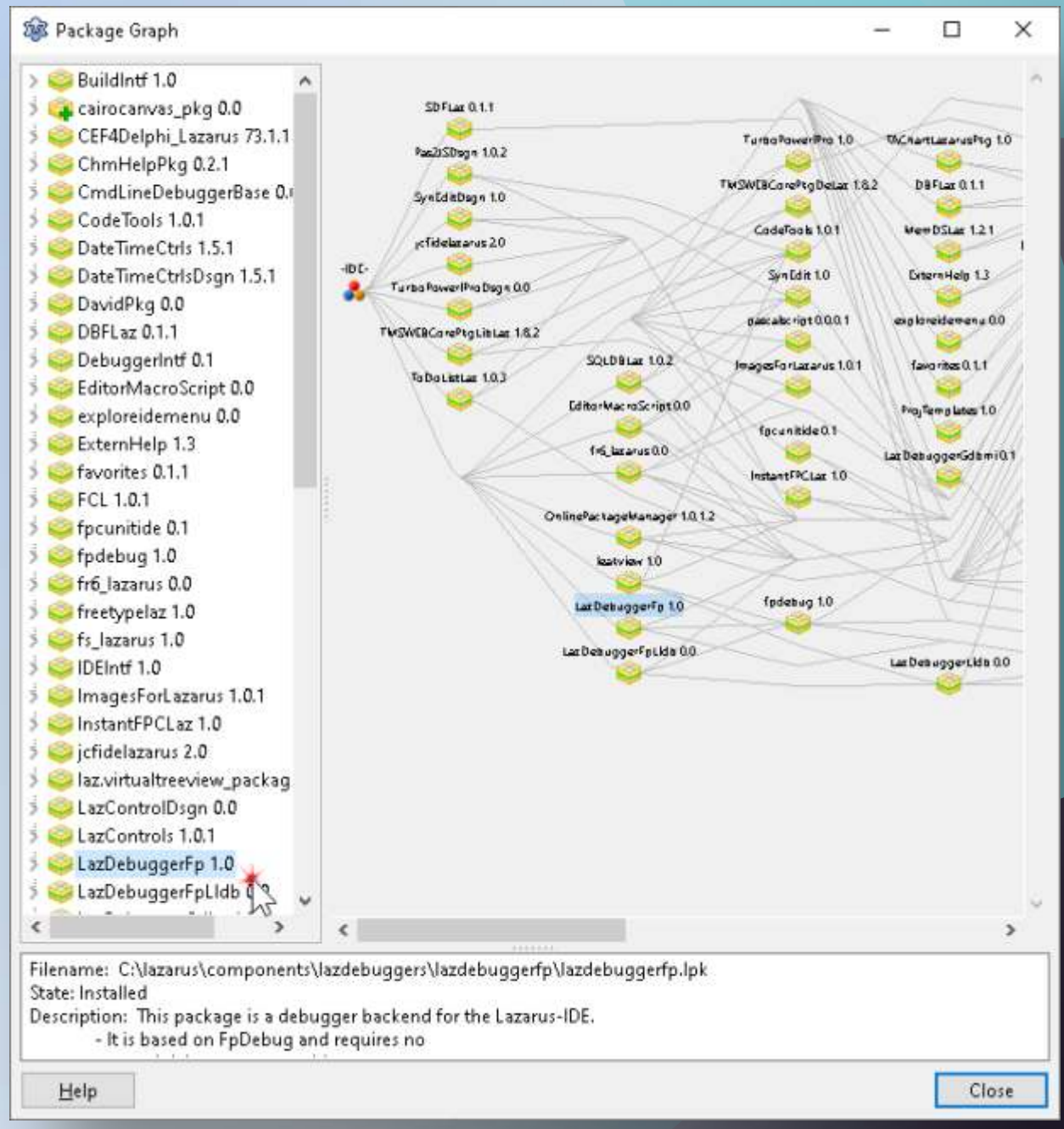



Figure 1: The Package Graph

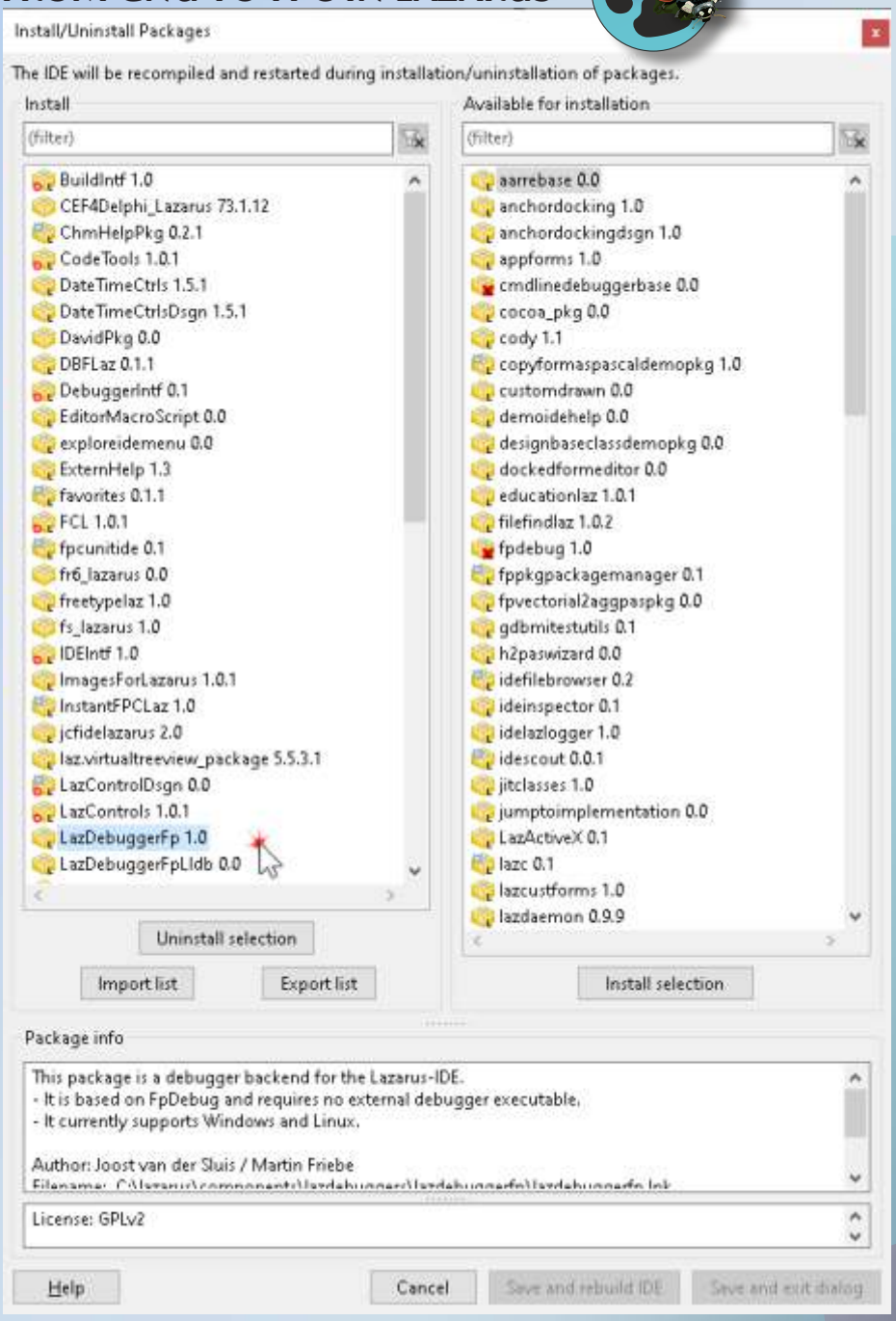



Figure 2: The Installing

In the **Menu → Package → Install/Uninstall Packages** install the package **`LazDebuggerFp`**, recompile **Lazarus.**
If the menu is not in the list at the right column,
it should be in the left column which means it is already installed.

After compilation,
you can switch the default debugger backend.
Go to **Tools àOptions** →**Debugger** → **Debugger backend** →click button **Add** there select in combobox **Debugger type and path** the debugger **FPDebug internal Dwarf-debugger.** Select this new created debugger backend in the upper combobox
Save this change with button **OK**. Now you use **FPDebug** as default debugger backend ( see figure 5/6 on page 4 of this article.

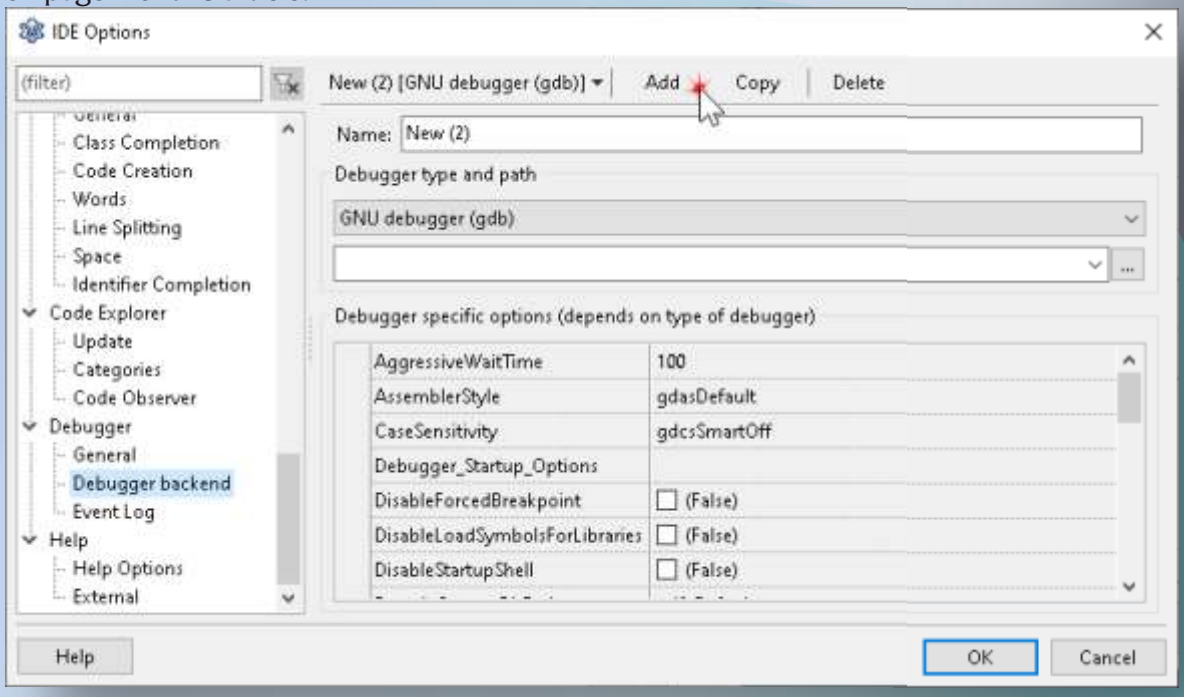



Figure 3: The Add button

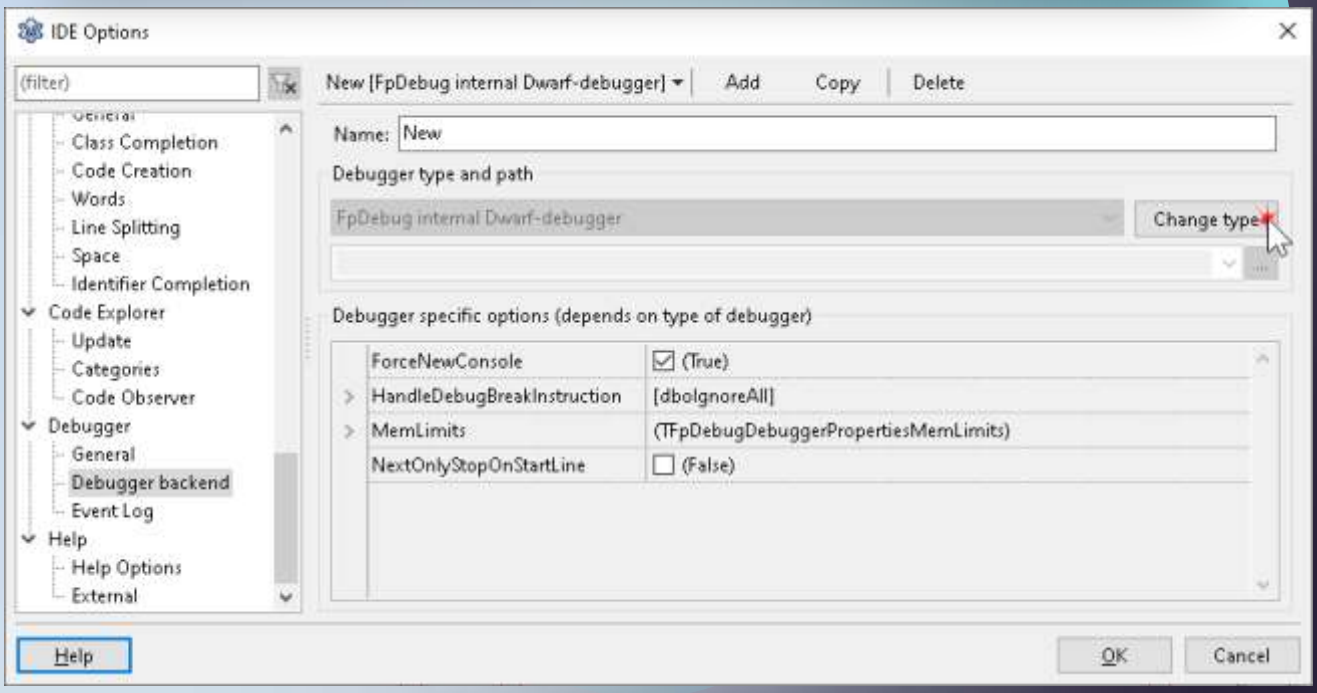



Figure 4: You can also change type

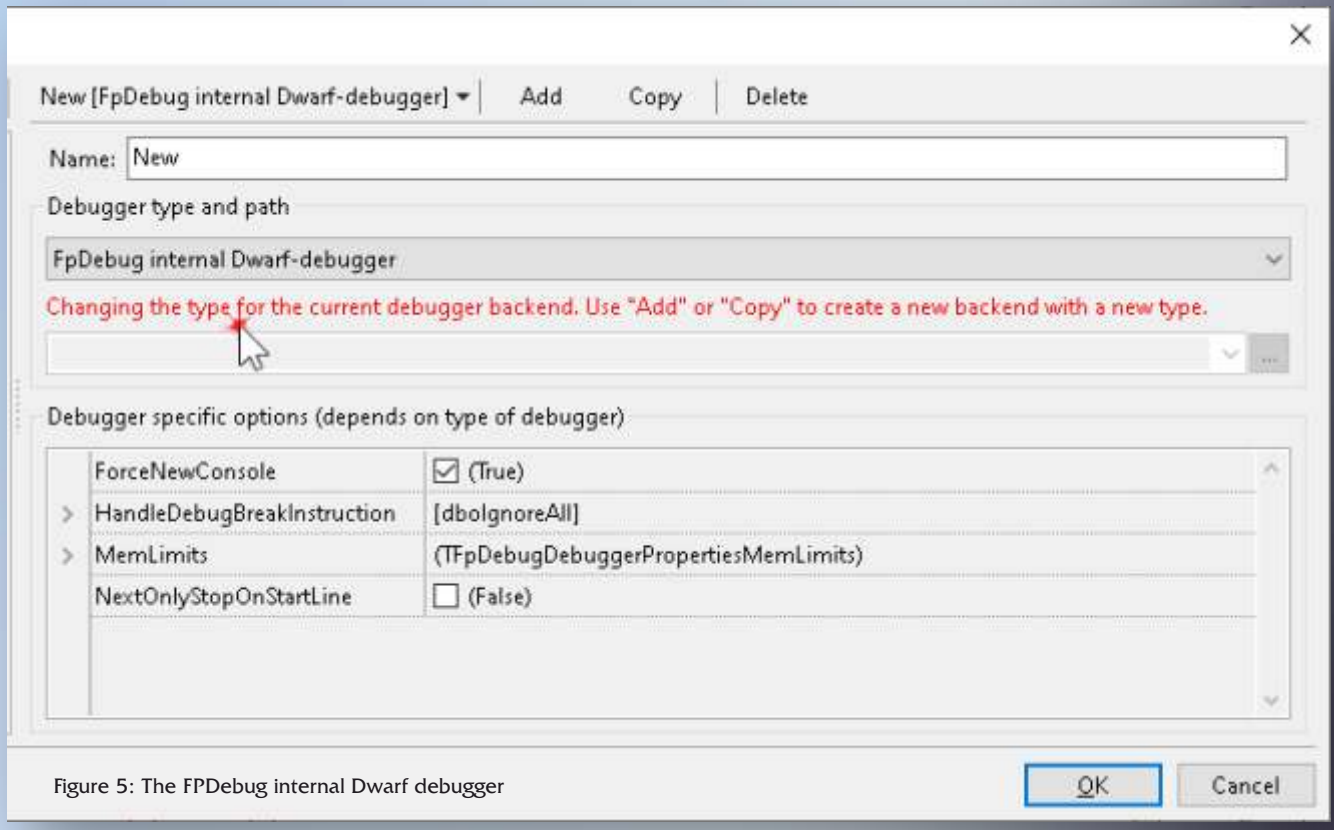



Figure 5: The FPDebug internal Dwarf debugger

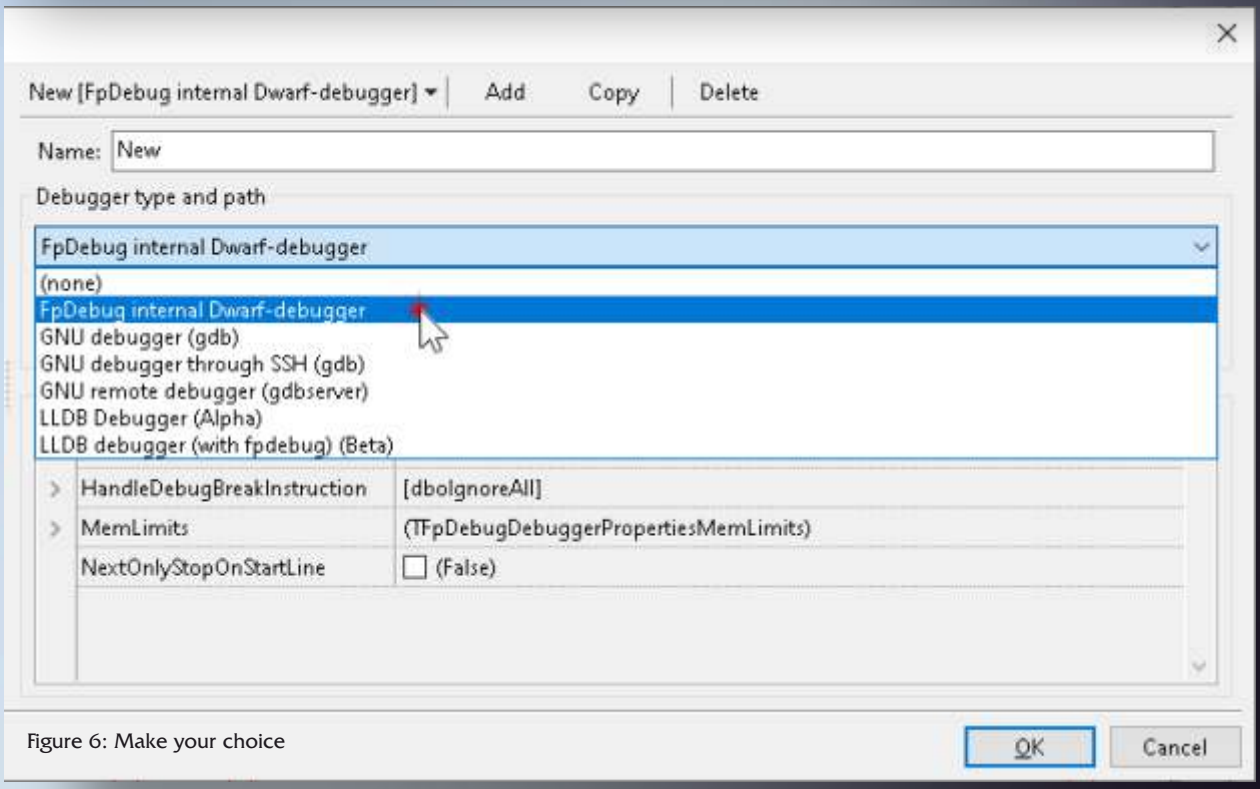
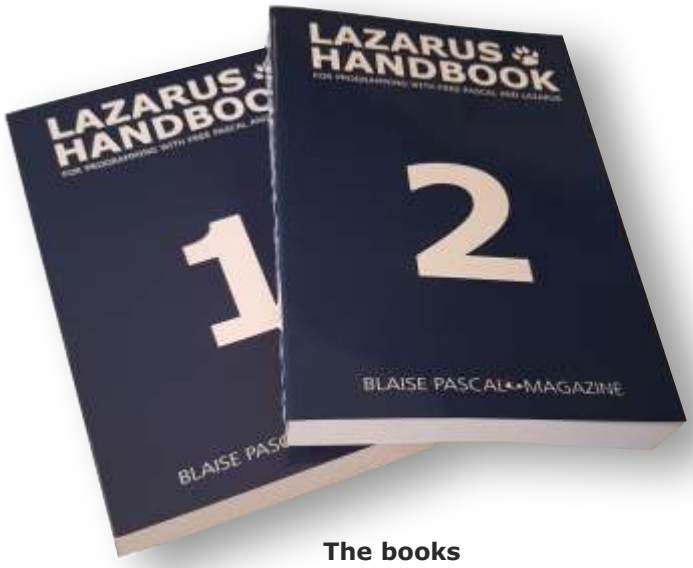



Figure 6: Make your choice

# SUMMERSALE

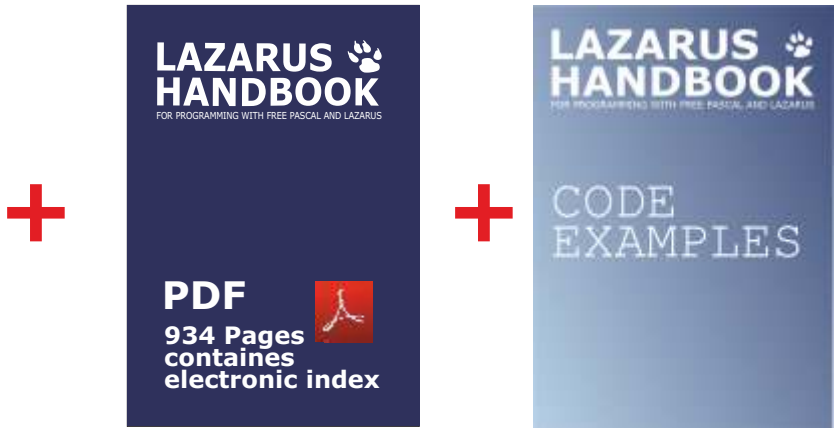



**The books**

# Sewn POCKET , almost thousand pages

**written by the makers of FPC and Lazarus**

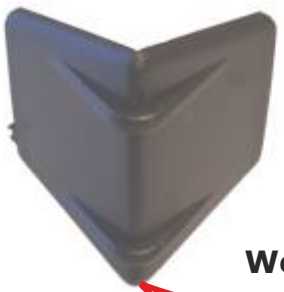
**934 Pages in two books 40 € (euro)**



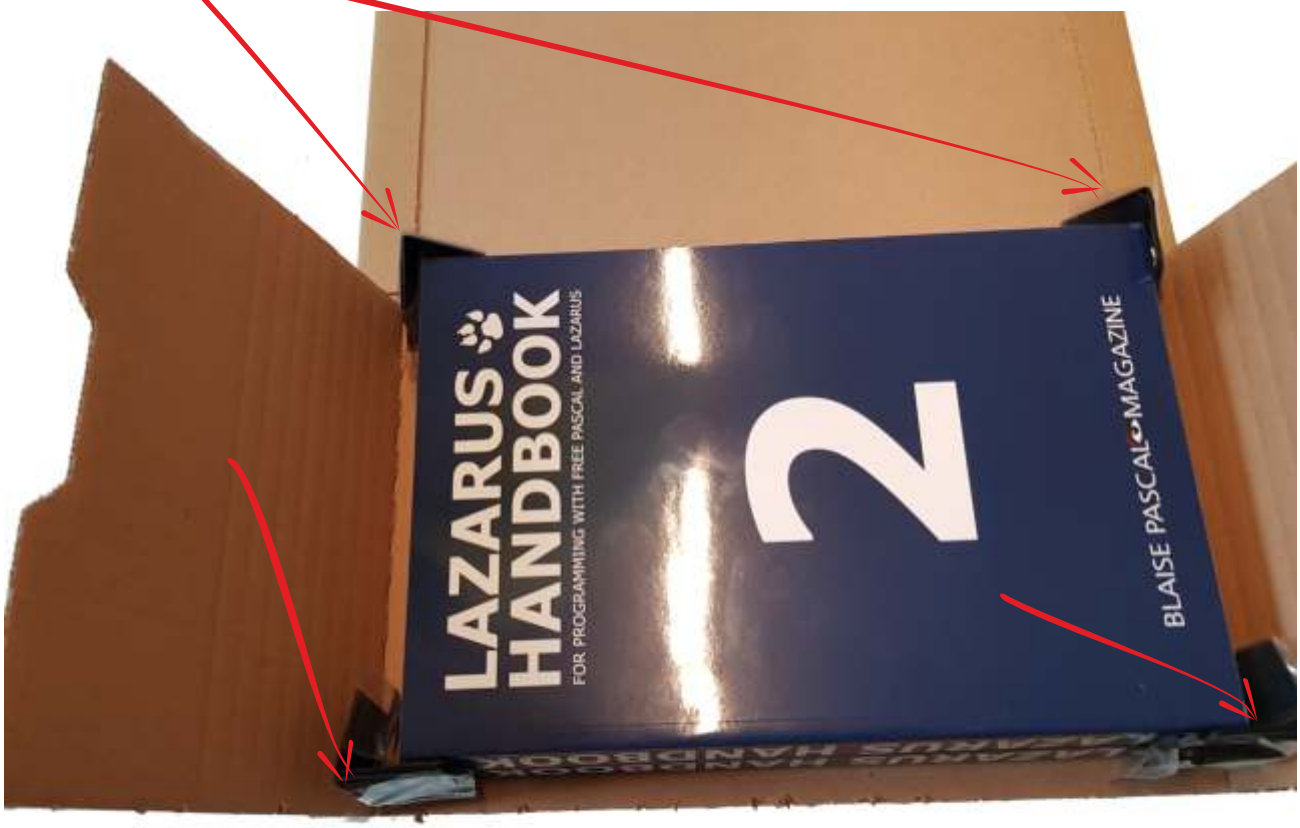

**Including the PDF and Code Examples**

https://www.blaisepascalmagazine.eu/product/lazarus-handbook-pocket/

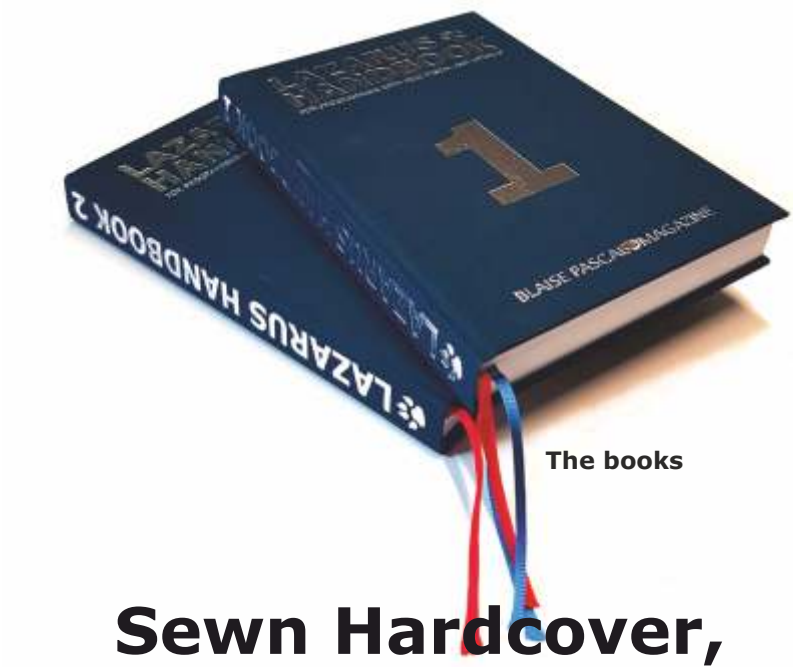
**We have a new service at our website, for shipping you can make three choices:**
**1.The shipping cost depending on which part of the world you want the book to be shipped:**
**Europe or the other countries: the World**



**We protect our shipments with extra plastic corners**



# Summersale

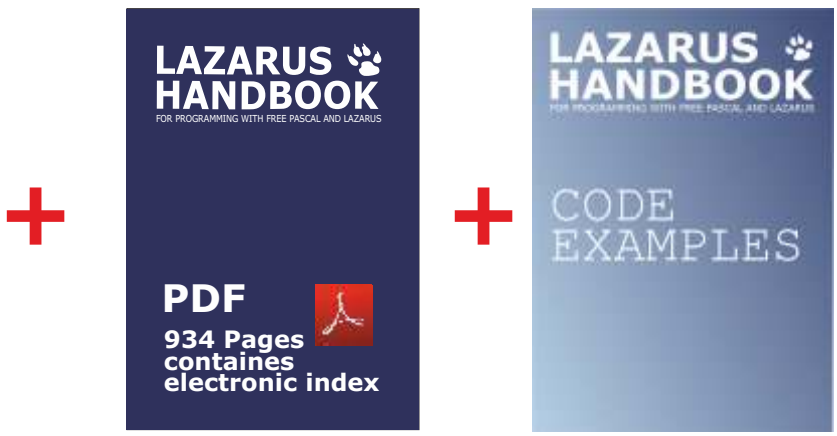



The books

## Sewn Hardcover, almost thousand pages

**written by the makers of FPC and Lazarus**

**934 Pages in two books 65 € (euro)**




**Including the PDF and Code Examples**

https://www.blaisepascalmagazine.eu/product/lazarus-handbook-hardcover/

**We have a new service at our website, for shipping you can make three choices:**
**1.The shipping cost depending on which part of the world you want the book to be shipped:**
**Europe or the other countries: the World**


This article has two mains items: learning to understand items about passwords and encryption and the second item is creating your own Password Creator app

A **password,** sometimes called a **passcode,** is secret data, typically a string of characters, usually used to confirm a user's identity.

Traditionally, passwords were expected to be memorized, but the large number of password-protected services that a typical individual accesses can make memorization of unique passwords for each service impractical. When the **claimant** successfully demonstrates knowledge of the password to the **verifier** through an established authentication protocol, the verifier is able to infer the claimant's identity.

In general, a password is an arbitrary string of characters including letters, digits, or other symbols. If the permissible characters are constrained to be numeric, the corresponding secret is sometimes called a **personal identification number (PIN).**
Despite its name, a password does not need to be an actual or real word; indeed, a non-word (in the dictionary sense) may be harder to guess, which is a desirable property of passwords. Using the terminology of the **NIST (*1)** Digital Identity Guidelines, the secret is held by a party called the **claimant** while the party verifying the identity of the claimant is called the **verifier.**
*1 (*NIST: National Institute of Standards and Technology*)
A memorized secret, consisting of a sequence of words or other text separated by spaces is sometimes called a passphrase. A passphrase is similar to a password in usage, but the former is generally longer for added security.

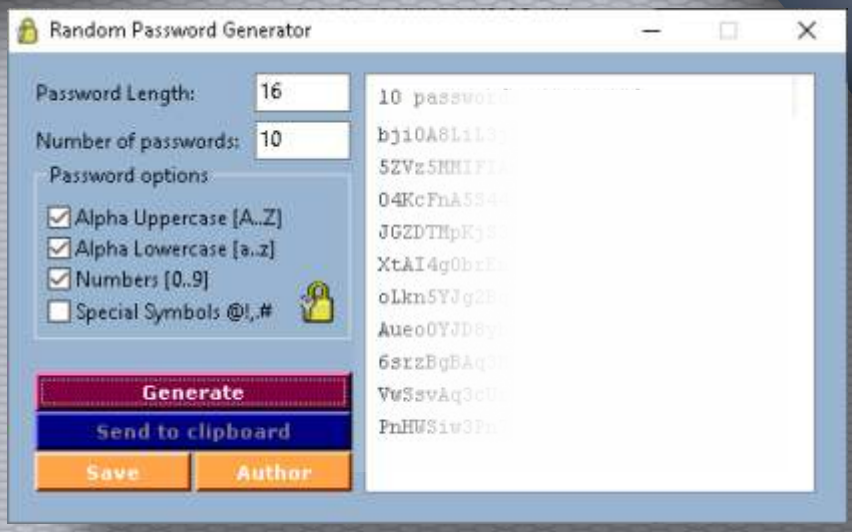



Figure 1: The program

WIKIPEDIA We have extensively been using WikiPedia for this artiicle

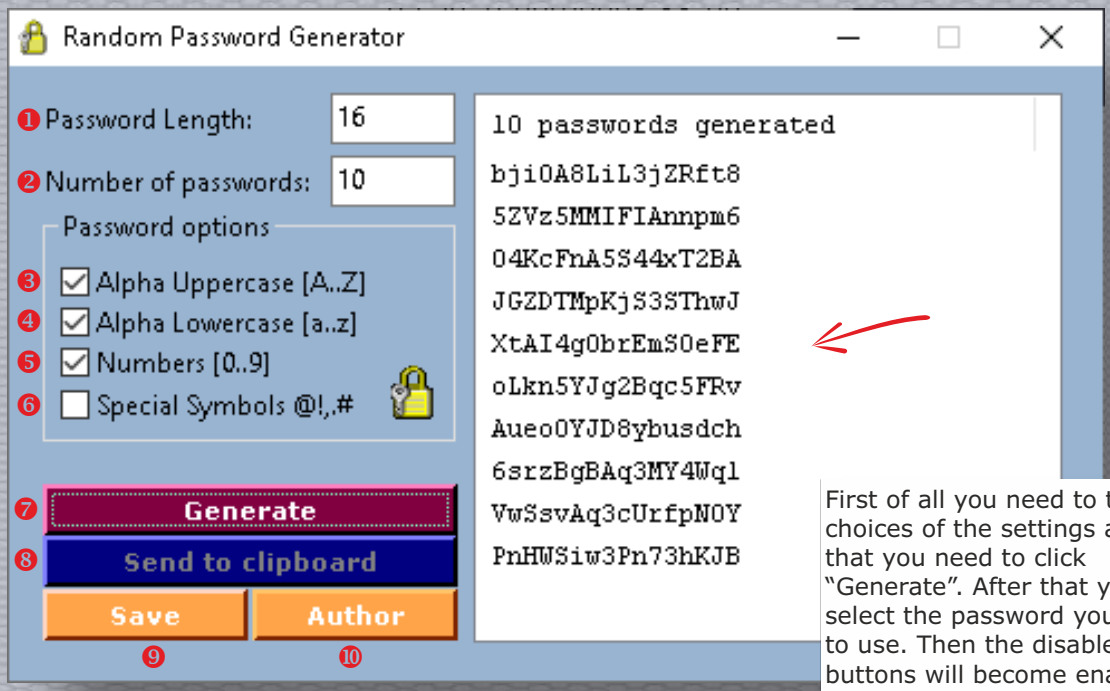



Figure 2:

❶ **Password Length**
It is advisable to set this number to at least 16, but better is 32.

❷ **Number of Passwords**
Ask as many as you wish. If you create a passphrase (a text that can easily be remembered you could easily add a Password in the middle of it. Don't recite a poem, that is easy to find, but maybe talk about your favourite food recipe that only you know.

❸ **Alpha Upper case**
If you check this it will be alphabetical and in CAPITALS

❹ **Alpha Lower case**
Just the same only lower case

❺ **Numbers**
This can be an extra mixture for your recipe

❻ **Special Symbols**
You can use them but better is longer…

❼ **Generate**
Create the passwords If you want to use a passphrase you can add the passwords. Maybe even three or fore groups...ore more

❽ **Send to clipboard**
There are several options:
if you select one
if you right-click

❾ **Save as File,** if you want to add it on your prepared USB Stick

❿ **About the initiating author**

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
 Classes, SysUtils, FileUtil, Forms, Clipbrd, Controls, Graphics, Dialogs,
 StdCtrls,  ExtCtrls, ComCtrls, Menus, HSButton;

const
 //Password mask consts
 TLowerAlpha = 'abcdefghijklmnopqrstuvwxyz';
 TUpperAlpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
 TNumber     = '0123456789';
 TSpecial    = '`!"?$?%^&*()"_-+={[}]:;@~#|\<,>./';

type

 { Tfrmmain }

 Tfrmmain = class(TForm)
  CheckGroup1: TCheckGroup;
...
...
 private
  { private declarations }
 public
  { public declarations }
 end;

var
 frmmain: Tfrmmain;
 rChar: integer;
 SelectedPW : String ;

function isNumber(buff: string): boolean;
function GetPassword(PasswordMask: string; pLength: integer): string;


 implementation

{$R *.lfm}

{ Tfrmmain }

function ListToString(LV: TListView): string;
var
 counter: integer;
 buff: string;
begin
 //Init var
 buff := '';
 //Loop tho the items in listview.
 for counter := 0 to LV.Items.Count - 1 do
 begin
  buff := buff + LV.Items[counter].Caption + #13 + #10;
 end;
 //Return result
 Result := buff;
end;
```

```
function isNumber(buff: string): boolean;
begin   //Check for empty string
   if buff = '' then
      begin
         Result := False;
         exit;
      end;
      try
         StrToInt(buff);
         Result := True;
      except
         Result := False;
      end;
end;

function GetPassword(PasswordMask: string; pLength: integer): string;
var  Counter: integer;  rPass: string;
begin

 rPass := '';

 for Counter := 1 to pLength do
 begin
  //Get randdom char from password set
  rChar := Random(Length(PasswordMask));
  //Build random password
  rPass := rPass + Copy(PasswordMask, rChar, 1);
 end;
 //Return password
 Result := rPass;
end;

procedure Tfrmmain.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
 Clipboard.AsText := '';
end;

procedure Tfrmmain.HSBtnMemoryClick(Sender: TObject);
begin
 If SelectedPW > '' then
 begin
  Clipboard.AsText := SelectedPW;
 end;
end;

procedure Tfrmmain.HSBtnGenerateClick(Sender: TObject);
var  pMask: string; Counter, pLen, pNum, integer; LItem: TListItem;
begin
 //Set defaults
 pMask := '';
 LstPws.Items.Clear;
 LstPws.Column[0].Caption := '0 passwords generated';
 HSBtnSave.Enabled:= True;

 //Check for vaild password length
 if not isNumber(txtLength.Text) then
 begin
  MessageDlg('Length is not a vaild integer.',
   mtWarning, [mbOK], 0);
  exit;
 end;
 //Check for vaid number of passwords
 if not isNumber(txtNum.Text) then
 begin
  MessageDlg('Number of passwords is not a vaild integer.',
   mtWarning, [mbOK], 0);
  exit;
 end;
```

→ This procedure is continued on the next page

Contiuation of: `procedure Tfrmmain.HSBtnGenerateClick(Sender: TObject);`

```
 //Assign the variables
 pLen := StrToInt(txtLength.Text);
 pNum := StrToInt(txtNum.Text);

 //Check for password length
 if pLen <= 0 then
 begin
  MessageDlg('Your password length needs to be greater then zero.',
   mtWarning, [mbOK], 0);
  exit;
 end;

 //Check number of passwords
 if pNum <= 0 then
 begin
  MessageDlg('The number of passwords needs to be greater than zero.',
   mtWarning, [mbOK], 0);
  exit;
 end;

 //Create password mask.
 if chkUpper.Checked then
 begin
  //Create upper alpha random password
  pMask := TUpperAlpha;
 end;

 if chkLower.Checked then
 begin
  //Create lowercase alpha random password
  pMask := pMask + TLowerAlpha;
 end;

 if chkNumber.Checked then
 begin
  //Create number random password
  pMask := pMask + TNumber;
 end;

 if chkSpecial.Checked then
 begin
  //Create special sybmols random password
  pMask := pMask + TSpecial;
 end;

 //Check password mask length
 if Length(pMask) = 0 then
 begin
  MessageDlg('Please select a password option.', mtInformation, [mbOK], 0);
  exit;
 end;
 //Set listview header text
 LstPws.Column[0].Caption := IntToStr(pNum) + ' '+'passwords generated';

 //Randomize
 Randomize;
 for Counter := 1 to pNum do
 begin
  //Add item to listview control
  LItem := LstPws.Items.Add;
  //Add random password as item
  LItem.Caption := GetPassword(pMask, pLen);
 end;
 //
end;
```

```
procedure Tfrmmain.HSBtnSaveClick(Sender: TObject);
var  T: TextFile;
begin  // Set the name of the file that will be created
   assignFile(T, SFD.FileName);
   LstPws.Column[0].Caption := '';

   // Use exceptions to catch errors (this is the default so not absolutely requried)
   {$I+}

   // Embed the file creation in a try/except block to handle errors gracefully
   try  // Create the file, write some text and close it.
        // Check if there is anything to be saved.
   if LstPws.Items.Count = 0
   then
      begin
         MessageDlg('Nothing to save.', mtInformation, [mbOK], 0);
         exit;
      end
   else
      begin
      //Setup save dialog props
         SFD.Title := 'Save Passwords';
         SFD.Filter := 'text|*.Txt';

      if SFD.Execute
      then
         begin
            //Save the list of passwords
            AssignFile(T, SFD.FileName);
            Rewrite(T);
            //Write the listview contents to the filename
            Write(T, ListToString(LstPws));
            CloseFile(T);
            HSBtnSave.Enabled:= False;
         end
      else
         exit;
      end;
   except  // If there was an error the reason can be found here
   on E: EInOutError do
   writeln('File handling error occurred. Details: ', E.ClassName, '/', E.Message);
   end;

   Clipboard.AsText := '';
   LstPws.Clear;
   LstPws.Refresh;
   HSBtnMemory.Enabled := False;
end;

procedure Tfrmmain.HSBtnAuthorClick(Sender: TObject);
var  msg: string;
begin  //Text for messagebox
   msg := self.Caption + ' v2.0' + #13 + #10 + ' originally developed by Ben Jones'
                       + #13 + #10 + 'and edited by Blaise Pascal Magazine September 2021';
   //Show our Messagebox
   MessageDlg(msg, mtInformation, [mbOK], 0);
end;

procedure Tfrmmain.LstPwsSelectItem(Sender: TObject; Item: TListItem; Selected: Boolean);
Var SelectedPW : String ;
begin
 If Selected then
 begin
  SelectedPW := Item.Caption ;
  HSBtnMemory.Enabled:= True;
  HSBtnSave.Enabled:= True;
  Clipboard.AsText := SelectedPW;
 end;
end;

end.
```

HISTORY AND FACTS

Passwords have been used since ancient times.
Sentries would challenge those wishing to enter an area to supply a password or watchword, and would only allow a person or group to pass if they knew the password. **Polybius** describes the system for the distribution of watchwords in the Roman military as follows:

> The way in which they secure the passing round of the **watchword** for the night is as follows: from the tenth **maniple** (three rows of 40 men) of each class of infantry and cavalry, the maniple which is encamped at the lower end of the street, a man is chosen who is relieved from guard duty, and he attends every day at sunset at the tent of the tribune, and receiving from him the watchword—that is a wooden tablet with the word inscribed on it – takes his leave, and on returning to his quarters passes on the watchword and tablet before witnesses to the commander of the next maniple, who in turn passes it to the one next to him. All do the same until it reaches the first maniples, those encamped near the tents of the tribunes. These latter are obliged to deliver the tablet to the tribunes before dark. So that if all those issued are returned, the tribune knows that the watchword has been given to all the maniples, and has passed through all on its way back to him. If any one of them is missing, he makes inquiry at once, as he knows by the marks from what quarter the tablet has not returned, and whoever is responsible for the stoppage meets with the punishment he merits. See figure 3


Figure 4: The Roman Seal

Passwords in military use evolved to include not just a password, but a password plus a counter password; for example in the opening days of the **Battle of Normandy,** paratroopers of the **U.S. 101st Airborne Division** used a password — *flash* — which was presented as a challenge, and answered with the correct response — *thunder*.
The challenge and response were changed every three days.
American paratroopers also famously used a device known as a "cricket" on D-Day in place of a password system as a temporarily unique method of identification:
one metallic click given by the device in lieu of a password was to be met by two clicks in reply.

Figure 3: Polybius

WIKIPEDIA We have extensively been using WikiPedia for this artiicle

Passwords have been used with computers since the earliest days of computing. The **Compatible Time-Sharing System (CTSS)**, an operating system introduced at **MIT (Massachusetts Institute of Technology)** in 1961, was the first computer system to implement password login.

CTSS had a LOGIN command that requested a user password.
"After typing PASSWORD, the system turns off the printing mechanism, if possible, so that the user may type in his password with privacy."
In the early 1970s, **Robert Morris** developed a system of storing login passwords in a hashed form (########) as part of the Unix operating system.
The system was based on a simulated **Hagelin rotor crypto machine**, and first appeared in 6th Edition Unix in 1974. A later version of his algorithm, known as crypt, used a 12-bit **Salt** (*see page 10*) and invoked a modified form of the DES algorithm 25 times to reduce the risk of pre-computed dictionary attacks. *The* **Data Encryption Standard (DES)** *is a symmetric-key algorithm for the encryption of digital data. Although its short key length of 56 bits makes it too insecure for applications, it has been highly influential in the advancement of cryptography.)*

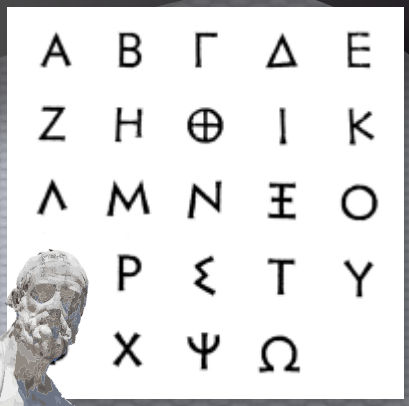




Figure 5: The original square used the Greek alphabet laid out left: With the modern Latin alphabet, this is the typical form (right):

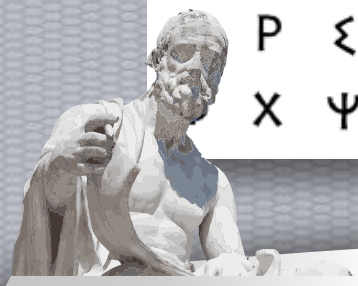


Figure 6: The Greek letters of a Polybius square

In modern times, user names and passwords are commonly used by people during a log in process that controls access to protected computer operating systems, mobile phones, cable TV decoders, Automated Teller Machines (ATMs), etc.
A typical computer user has passwords for many purposes: logging into accounts, retrieving e-mail, accessing applications, databases, networks, web sites, and even reading the morning newspaper online.
**Polybius** was responsible for a useful tool in telegraphy that allowed letters to be easily signalled using a numerical system, called "the Polybius square" (*See Figure* 6). This idea also lends itself to cryptographic manipulation and **steganography*2.** Modern implementations of the Polybius square, at least in Western European languages such as English, Spanish, French, German, and Italian, generally use the Roman Alphabet in which those languages are written. However, Polybius himself was writing in Greek, and would have implemented his cipher square in the Greek Alphabet. Both versions are shown here.

## HISTORY

*2(**Steganography** *is the practice of concealing a message within another message or a physical object. In computing/electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video. The word steganography comes from Greek "steganographia", which combines the words steganós, meaning "covered or concealed", and -graphia meaning "writing".*

*The first recorded use of the term was in 1499 by* **Johannes Trithemius** *in his* **Steganographia,** *a treatise on cryptography and steganography, disguised as a book on magic.*
*Generally, the hidden messages appear to be (or to be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter.*

*The advantage of* **steganography** *over* **cryptography** *alone is that the intended secret message does not attract attention to itself as an object of scrutiny.*
*Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.*
*Whereas* **cryptography** *is the practice of protecting the contents of a message alone,* **steganography** *is concerned with concealing the fact that a secret message is being sent and its contents.*

**steganography** *includes the concealment of information within computer files.*
*In digital steganography, electronic communications may include* steganographic *coding inside of a transport layer, such as a document file, image file, program, or protocol.*
*Media files are ideal for steganographic transmission because of their large size.*
*For example, a sender might start with an innocuous image file and adjust the color of every hundredth pixel to correspond to a letter in the alphabet. The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.*
*See the images below:*

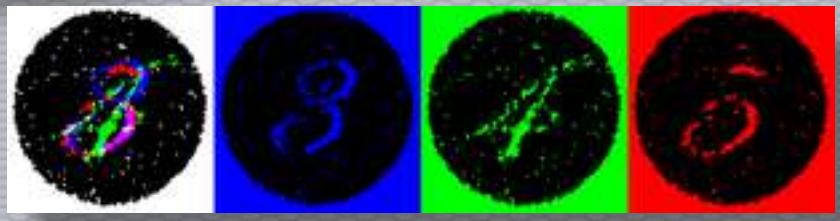


Figure 7: The same image viewed by white, blue, green, and red lights reveals different hidden numbers.

## SALTING

In cryptography, a **salt** is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase.
**Salt** are used to safeguard passwords in storage.
Historically, only a cryptographic hash function of the password was stored on a system, but over time, additional safeguards were developed to protect against duplicate or common passwords being identifiable (*as their hashes are identical*).
Salting is one such protection.

A new **salt** is randomly generated for each password.
Typically, the **salt** and the password (*or its version after key stretching*) are concatenated and fed to a cryptographic hash function, and the output hash value (but not the original password) is stored with the **salt** in a database.

Hashing allows for later authentication without keeping and therefore risking exposure of the plaintext password if the authentication data store is compromised.
Note that due to this, salts don't need to be encrypted or stored separately from the hashed password itself, because even if an attacker has access to the database with the hash values and the salts, the correct use of salts will hinder common attacks.
**Salts** defend against attacks that use precomputed tables (*e.g. rainbow tables *3*) as they can make the size of table needed for a successful attack prohibitively large without burdening users.
Since salts differ from one another, they also protect redundant (e.g. commonly-used, re-used) passwords as different **salted** hashes are created for different instances of the same password.
Cryptographic **salts** are broadly used in many modern computer systems, from Unix system credentials to Internet security. **Salts** are closely related to the concept of a **cryptographic nonce.**

For example, ordering a product using an e-commerce site typically uses a nonce to assign originality to a purchase. Without this, an attacker could potentially replay the encrypted information as many times as desired to continue placing orders under the same name and purchase information.

**3 (A* **rainbow table** *is a precomputed table for caching the output of cryptographic hash functions, usually for cracking password hashes. Tables are usually used in recovering a key derivation function (or credit card numbers, etc.) up to a certain length consisting of a limited set of characters. It is a practical example of a space–time trade-off, using less computer processing time and more storage than a brute-force attack which calculates a hash on every attempt, but more processing time and less storage than a simple key derivation function with one entry per hash.)*
**Use of a key derivation that employs a salt makes this attack infeasible.**

**DES** (**Data Encryption Standard**) has been considered insecure right from the start because of the feasibility of brute-force attacks. Such attacks have been demonstrated in practice and are now available on the market as a service. As of 2008, the best analytical attack is linear cryptanalysis, which requires 243 known plain texts and has a time complexity of 239–43 (June 2001).

In cryptography, **encryption** is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext.
Ideally, only authorized parties can decipher a ciphertext back to plaintext and access the original information. Encryption does not itself prevent interference but denies the intelligible content to a would-be interceptor.

For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is possible to decrypt the message without possessing the key but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

Historically, various forms of encryption have been used to aid in cryptography.
Early encryption techniques were often utilized in military messaging.
Since then, new techniques have emerged and become commonplace in all areas of modern computing. Modern encryption schemes utilize the concepts of public-key and symmetric-key.

**Modern encryption techniques ensure security because modern computers are inefficient at cracking the encryption.**

- **One of the earliest forms of encryption** is **symbol replacement**, which was first found in the **tomb of Khnumhotep II**, who lived in 1900 BC Egypt.
Symbol replacement encryption is "non-standard," which means that the symbols require a cipher or key to understand.
This type of early encryption was used throughout Ancient Greece and Rome for military purposes.
One of the most famous military encryption developments was the **Caesar Cipher**, which was a system in which a letter in normal text is shifted down a fixed number of positions down the alphabet to get the encoded letter.
A message encoded with this type of encryption could be decoded with the fixed number on the **Caesar Cipher**.

- Around 800 AD, Arab mathematician **Al-Kindi** developed the technique of frequency analysis – which was an attempt to systematically crack Caesar ciphers.
This technique looked at the frequency of letters in the encrypted message to determine the appropriate shift.
This technique was rendered ineffective after the creation of the **Polyalphabetic cipher** by **Leone Alberti** in 1465, which incorporated different sets of languages. In order for frequency analysis to be useful, the person trying to decrypt the message would need to know which language the sender chose.

- 9th–20th century
Around 1790, Thomas Jefferson (Former American president) theorised a cipher to encode and decode messages in order to provide a more secure way of military correspondence.
The cipher, known today as the **Wheel Cipher or the Jefferson Disk**, although never actually built, was theorized as a spool that could jumble an English message up to 36 characters.
The message could be decrypted by plugging in the jumbled message to a receiver with an identical cipher.

A similar device to the **Jefferson Disk**, the **M-94**, was developed in 1917 independently by US Army Major **Joseph Mauborne**. This device was used in U.S. military communications until 1942. *See Figure: 8*


Figure 8: The Jefferson Disk

*The* **M-94** *was a piece of cryptographic equipment used by the United States Army, consisting of several lettered discs arranged as a cylinder. It was also employed by the US Navy, under the name* **CSP 488**.
*The device was conceived by* **Colonel Parker Hitt** *and then developed by* **Major Joseph Mauborgne** *in 1917; based on a system invented by* **Thomas Jefferson** *and* **Etienne Bazeries.** *Officially adopted in 1922, it remained in use until circa 1942, when it was replaced by more complex and secure electromechanical rotor machines, particularly the* **M-209.** *See below Figure 9)*
In World War II, the Axis powers used a more advanced version of the M-94 called the **Enigma Machine.**
The **Enigma Machine** was more complex because unlike the Jefferson Wheel and the M-94, each day the jumble of letters switched to a completely new combination.
Each day's combination was only known by the Axis, so many thought the only way to break the code would be to try over 17,000 combinations within 24 hours.
The Allies used computing power to severely limit the number of reasonable combinations they needed to check every day, leading to the breaking of the Enigma Machine.

## MODERN

Today, encryption is used in the transfer of communication over the Internet for security and commerce. As computing power continues to increase, computer encryption is constantly evolving to prevent attacks.

## ENCRYPTION IN CRYPTOGRAPHY

In the context of cryptography, encryption serves as a mechanism to ensure confidentiality. Since data may be visible on the Internet, sensitive information such as passwords and personal communication may be exposed to potential interceptors. The process of encrypting and decrypting messages involves keys. The two main types of keys in cryptographic systems are symmetric-key and public-key (also known as asymmetric-key). Many complex cryptographic algorithms often use simple modular arithmetic in their implementations.

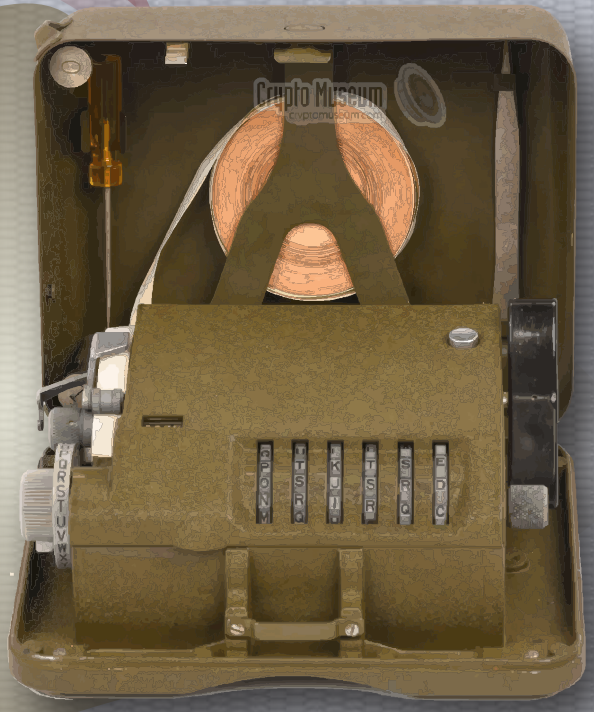



Figure 9: The M 209

**Symmetric key**

In symmetric-key schemes, the encryption and decryption keys are the same. Communicating parties must have the same key in order to achieve secure communication. The German Enigma Machine utilized a new symmetric-key each day for encoding and decoding messages.

**Public key**

In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read. Public-key encryption was first described in a secret document in 1973; beforehand, all encryption schemes were symmetric-key (also called private-key). Although published subsequently, the work of **Diffie** and **Hellman** was published in a journal with a large readership, and

**RSA**

**RSA (Rivest–Shamir–Adleman)** is another notable public-key cryptosystem.
Created in 1978, it is still used today for applications involving digital signatures.
Using number theory, the RSA algorithm selects two prime numbers, which help generate both the encryption and decryption keys.
A publicly available public-key encryption application called Pretty Good Privacy ( *PGP* ) was written in 1991 by Phil Zimmermann, and distributed free of charge with source code. PGP was purchased by Symantec in 2010 and is regularly updated.

**Uses**

Encryption has long been used by militaries and governments to facilitate secret communication.
It is now commonly used in protecting information within many kinds of civilian systems.
For example, the Computer Security Institute reported that in 2007, 71% of companies surveyed utilized encryption for some of their data in transit, and 53% utilized encryption for some of their data in storage.
Encryption can be used to protect data "at rest", such as information stored on computers and storage devices (e.g. USB flash drives). In recent years, there have been numerous reports of confidential data, such as customers' personal records, being exposed through loss or theft of laptops or backup drives; encrypting such files at rest helps protect them if physical security measures fail.
**Digital rights management systems**, which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering (*see also copy protection*), is another somewhat different example of using encryption on data at rest.
Encryption is also used to protect data in transit, for example data being transferred via networks (e.g. the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in transit being intercepted in recent years. Data should also be encrypted when transmitted across networks in order to protect against eavesdropping of network traffic by unauthorized users.

### Data erasure

Conventional methods for permanently deleting data from a storage device involve overwriting the device's whole content with zeros, ones, or other patterns – a process which can take a significant amount of time, depending on the capacity and the type of storage medium.
**Cryptography** offers a way of making the erasure almost instantaneous. This method is called crypto-shredding. An example implementation of this method can be found on iOS devices, where the cryptographic key is kept in a dedicated 'effaceable storage'. Because the key is stored on the same device, this setup on its own does not offer full privacy or security protection if an unauthorized person gains physical access to the device.

### Limitations

**Encryption** is used in the 21st century to protect digital data and information systems.
As computing power increased over the years, encryption technology has only become more advanced and secure. However, this advancement in technology has also exposed a potential limitation of today's encryption methods.
The length of the encryption key is an indicator of the strength of the encryption method.
For example, the original encryption key, **DES(Data Encryption Standard)**, was 56 bits, meaning it had 2^56 combination possibilities. With today's computing power, a 56-bit key is no longer secure, being vulnerable to hacking by brute force attack.

Today the standard of modern encryption keys is up to 2048 bit with the **RSA** system.
**Decrypting a 2048 bit encryption key is nearly impossible in light of the number of possible combinations.**
However, **quantum computing is threatening to change this secure nature**.
Quantum computing utilizes properties of quantum mechanics in order to process large amounts of data simultaneously. Quantum computing has been found to achieve computing speeds thousands of times faster than today's supercomputers.
This computing power presents a challenge to today's encryption technology. For example, **RSA encryption** utilizes the multiplication of very large prime numbers to create a semiprime number for its public key. Decoding this key without its private key requires this semiprime number to be factored in, which can take a very long time to do with modern computers. It would take a supercomputer anywhere between weeks to months to factor in this key.
However, quantum computing can use quantum algorithms to factor this semiprime number in the same amount of time it takes for normal computers to generate it. This would make all data protected by current public-key encryption vulnerable to quantum computing attacks.
**Other encryption techniques like elliptic curve cryptography and symmetric key encryption are also vulnerable to quantum computing.**
While quantum computing could be a threat to encryption security in the future, quantum computing as it currently stands is still very limited. Quantum computing currently is not commercially available, cannot handle large amounts of code, and only exists as computational devices, not computers. Furthermore, quantum computing advancements will be able to be utilized in favor of encryption as well. The National Security Agency (NSA) is currently preparing post-quantum encryption standards for the future.
**Quantum encryption promises a level of security that will be able to counter the threat of quantum computing.**

### Attacks and countermeasures

Encryption is an important tool but is not sufficient alone to ensure the security or privacy of sensitive information throughout its lifetime. **Most applications of encryption protect information only at rest or in transit**, leaving sensitive data in clear text and potentially vulnerable to improper disclosure during processing, such as by a cloud service for example.
**Homomorphic encryption*4** and **secure multi-party computation *5** are emerging techniques to compute on encrypted data; these techniques are general and **Turing complete** *6 but incur high computational and/or communication costs.

***4(Homomorphic encryption**
*is a form of encryption that permits users to perform computations on its encrypted data without first decrypting it. These resulting computations are left in an encrypted form which, when decrypted, result in an identical output to that produced had the operations been performed on the unencrypted data.*
*Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted.*

*For sensitive data, such as* **health care information**, *homomorphic encryption can be used to enable new services by removing privacy barriers inhibiting data sharing or increase security to existing services. For example, predictive analytics in health care can be hard to apply via a third party service provider due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished.*
**Moreover, even if the service provider's system is compromised, the data would remain secure.**)

***5(Secure multi-party computation**
**-** *also known as secure computation, multi-party computation (MPC), or privacy-preserving computation - is a subfield of cryptography with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.*
*Unlike traditional cryptographic tasks, where cryptography assures security and integrity of communication or storage and the adversary is outside the system of participants (an eavesdropper on the sender and receiver), the cryptography in this model protects participants' privacy from each other.***)**

***6(Turing-complete**

*In computability theory, a system of data-manipulation rules (such as a computer's instruction set, a programming language, or a cellular automation) is said to be Turing-complete or computationally universal if it can be used to simulate any* **Turing machine**. *This means that this system is able to recognize or decide other data-manipulation rule sets. Turing completeness is used as a way to express the power of such a data-manipulation rule set. Virtually all programming languages today are Turing-complete. The concept is named after English mathematician and computer scientist Alan Turing.* )

In response to encryption of data at rest, cyber-adversaries have developed new types of attacks. These more recent threats to encryption of data at rest include cryptographic attacks, stolen ciphertext attacks, attacks on encryption keys, insider attacks, data corruption or integrity attacks, data destruction attacks, and ransomware attacks. Data fragmentation and **active defense *7** data protection technologies attempt to counter some of these attacks, by distributing, moving, or mutating ciphertext so it is more difficult to identify, steal, corrupt, or destroy.

***7(Active defense**

*can refer to a defensive strategy in the military or cybersecurity arena.*
*In the cybersecurity arena, active defence may mean "asymmetric defences," namely defences that increase costs to cyber-adversaries by reducing costs to cyber-defenders. For example, an active defence data protection strategy invented by CryptoMove leverages dynamic data movement, distribution, and re-encryption to make data harder to attack, steal, or destroy.***)**

**Integrity protection of ciphertexts**

Encryption, by itself, can protect the confidentiality of messages, but other techniques are still needed to protect the integrity and authenticity of a message; for example, verification of a message authentication code (MAC) or a digital signature.
Authenticated encryption algorithms are designed to provide both encryption and integrity protection together.
Standards for cryptographic software and hardware to perform encryption are widely available, but successfully using encryption to ensure security may be a challenging problem.
A single error in system design or execution can allow successful attacks. Sometimes an adversary can obtain unencrypted information without directly undoing the encryption.
*See for example traffic analysis, TEMPEST, or Trojan horse.*

Integrity protection mechanisms such as MACs and digital signatures must be applied to the ciphertext when it is first created, typically on the same device used to compose the message, to protect a message end-to-end along its full transmission path;
otherwise, any node between the sender and the encryption agent could potentially tamper with it.
Encrypting at the time of creation is only secure if the encryption device itself has correct keys and has not been tampered with. If an endpoint device has been configured to trust a root certificate that an attacker controls, for example, then the attacker can both inspect and tamper with encrypted data by performing a man-in-the-middle attack anywhere along the message's path.
The common practice of **TLS interception** by network operators represents a controlled and institutionally sanctioned form of such an attack, but countries have also attempted to employ such attacks as a form of control and censorship.

## CIPHERTEXT LENGTH AND PADDING

Even when encryption correctly hides a message's content, and it cannot be tampered with at rest or in transit, a “message's length” is a form of metadata that can still leak sensitive information about the message.
*(**Metadata** *is "data that provides information about other data". In other words, it is "data about data". Many distinct types of metadata exist, including descriptive metadata, structural metadata, administrative metadata, reference metadata, statistical metadata and legal metadata.)*

For example, the well-known **CRIME** and **BREACH** attacks against **HTTPS** were side-channel attacks that relied on information leakage, via the length of encrypted content.

Traffic analysis is a broad class of techniques that often employs message lengths to infer sensitive implementation about traffic flows, by aggregating information about a large number of messages.

Padding a message's payload, before encrypting it can help obscure the cleartext's true length, at the cost of increasing the ciphertext's size and introducing or increasing bandwidth overhead.

Messages may be padded randomly or deterministically, with each approach having different trade offs.

Encrypting and padding messages to form padded uniform random blobs or **PURBs (Padded Uniform Random Blob*)**, is a practice guaranteeing that the cipher text leaks no metadata about its cleartext's content, and leaks asymptotically minimal information via its length.

***(Binary Large Object)**

## PASSWORD STRENGTH

is a measure of the effectiveness of a password against guessing or brute-force attacks.
In its usual form, it estimates how many trials an attacker who does not have direct access to the password would need, on average, to guess it correctly. The strength of a password is a function of length, complexity, and unpredictability.

Using strong passwords lowers overall risk of a security breach, but strong passwords do not replace the need for other effective security controls.
The effectiveness of a password of a given strength is strongly determined by the design and implementation of the factors (knowledge, ownership, inherence).

The rate at which an attacker can submit guessed passwords to the system is a key factor in determining system security. Some systems impose a time-out of several seconds after a small number (e.g. three) of failed password entry attempts.
In the absence of other vulnerabilities, such systems can be effectively secured with relatively simple passwords. However, the system must store information about the user's passwords in some form and if that information is stolen, say by breaching system security, the user's passwords can be at risk.

In 2019, the **United Kingdom's NCSC** analysed public databases of breached accounts to see which words, phrases and strings people used. **Top of the list was 123456,** appearing in more than 23 million passwords. The second-most popular string, **123456789,** was not much harder to crack, while the top five included **"qwerty"**, **"password"** and **1111111**.

## PASSWORD CREATION

Passwords are created either automatically (using randomizing equipment) or by a human; the latter case is more common. While the strength of randomly chosen passwords against a brute-force attack can be calculated with precision, determining the strength of human-generated passwords is challenging.[4]

Typically, humans are asked to choose a password, sometimes guided by suggestions or restricted by a set of rules, when creating a new account for a computer system or Internet Web site. Only rough estimates of strength are possible since humans tend to follow patterns in such tasks, and those patterns can usually assist an attacker.[5] In addition, lists of commonly chosen passwords are widely available for use by password guessing programs. Such lists include the numerous online dictionaries for various human languages, breached databases of plaintext, and hashed passwords from various online business and social accounts, along with other common passwords. All items in such lists are considered weak, as are passwords that are simple modifications of them.

Although random password generation programs are available nowadays which are meant to be easy to use, they still usually generate random, hard to remember passwords, oftentimes resulting in people choosing to implement their own. However, this is inherently insecure because the person's lifestyles, entertainment preferences, and other key individualistic qualities usually come into play to influence the choice of password, while the prevalence of online social media has made obtaining information about people much easier.

## PASSWORD GUESS VALIDATION

Systems that use passwords for authentication must have some way to check any password entered to gain access. If the valid passwords are simply stored in a system file or database, an attacker who gains sufficient access to the system will obtain all user passwords, giving the attacker access to all accounts on the attacked system and possibly other systems where users employ the same or similar passwords.
One way to reduce this risk is to store only a cryptographic hash of each password instead of the password itself. Standard cryptographic hashes, such as the **Secure Hash Algorithm (SHA)** series, are very hard to reverse, so an attacker who gets hold of the hash value cannot directly recover the password.
However, knowledge of the hash value lets the attacker quickly test guesses offline. Password cracking programs are widely available that will test a large number of trial passwords against a purloined cryptographic hash.

Improvements in computing technology keep increasing the rate at which guessed passwords can be tested. For example, in 2010, the **Georgia Tech Research Institute** developed a method of using **GPGPU** to crack passwords much faster. **Elcomsoft** invented the usage of common graphic cards for quicker password recovery in August 2007 and soon filed a corresponding patent in the US.

By 2011, commercial products were available that claimed the ability to test up to 112,000 passwords per second on a standard desktop computer, using a high-end graphics processor for that time. Such a device will crack a six-letter single-case password in one day.
**Note** that the work can be distributed over many computers for an additional speedup proportional to the number of available computers with comparable GPUs**.**
**Special key stretching hashes are available that take a relatively long time to compute, reducing the rate at which guessing can take place.**
Although it is considered best practice to use key stretching, many common systems do not.

Another situation where quick guessing is possible is when the password is used to form a **cryptographic key**. In such cases, an attacker can quickly check to see if a guessed password successfully decodes encrypted data.
For example, one commercial product claims to test 103,000 WPA PSK passwords per second.

If a password system only stores the hash of the password, an attacker can pre-compute hash values for common passwords variants and for all passwords shorter than a certain length, allowing very rapid recovery of the password once its hash is obtained. Very long lists of pre-computed password hashes can be efficiently stored using rainbow tables. **This method of attack can be foiled by storing a random value, called a cryptographic salt*, along with the hash.**
The **salt** is combined with the password when computing the hash, so an attacker precomputing a rainbow table would have to store for each password its hash with every possible **salt*** value. **This becomes infeasible if the salt has a big enough range, say a 32-bit number.** Unfortunately, many authentication systems in common use do not employ salts and rainbow tables are available on the Internet for several such systems. * see page XX of this article for explanation.

## ENTROPY AS A MEASURE OF PASSWORD STRENGTH

It is usual in the computer industry to specify password strength in terms of **information entropy,** which is measured in bits and is a concept from information theory.
Instead of the number of guesses needed to find the password with certainty, the base-2 logarithm of that number is given, which is commonly referred to as the number of "entropy bits" in a password, though this is not exactly the same quantity as information entropy.

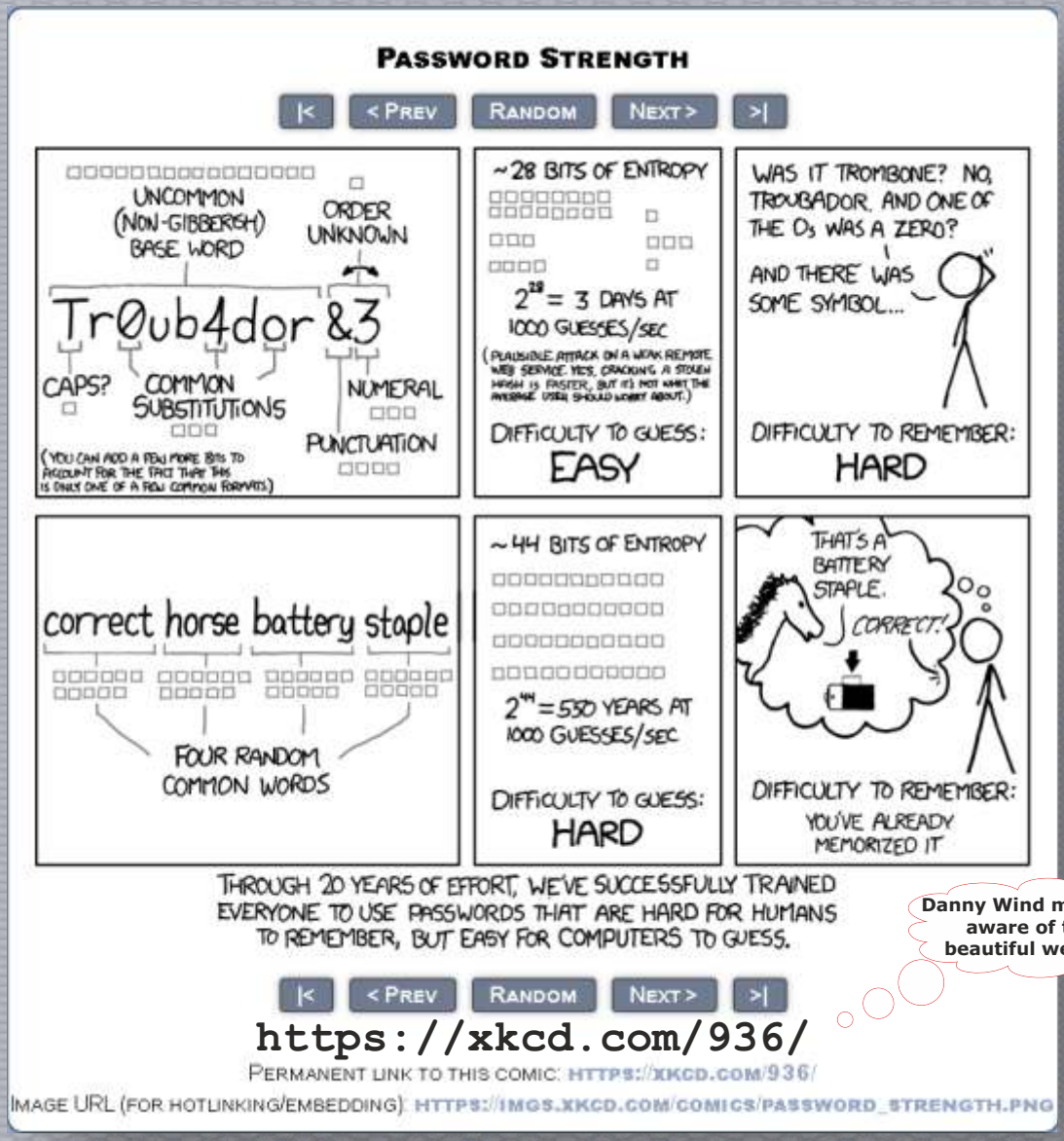
A password with an entropy of 42 bits calculated in this way would be as strong as a string of 42 bits chosen randomly, for example by a fair coin toss.
Put another way, a password with an entropy of 42 bits would require 242 (4,398,046,511,104) attempts to exhaust all possibilities during a brute force search.

Thus, increasing the entropy of the password by one bit doubles the number of guesses required, making an attacker's task twice as difficult. On average, an attacker will have to try half the possible number of passwords before finding the correct one.

RANDOM PASSWORDS

Random passwords consist of a string of symbols of specified length taken from some set of symbols using a random selection process in which each symbol is equally likely to be selected.
The symbols can be individual characters from a character set (e.g., the ASCII character set), syllables designed to form pronounceable passwords, or even words from a word list (*thus forming a passphrase*).

The strength of random passwords depends on the actual entropy of the underlying number generator; however, these are often not truly random, but pseudorandom.
Many publicly available password generators use random number generators found in programming libraries that offer limited entropy.
However most modern operating systems offer cryptographically strong random number generators that are suitable for password generation.
It is also possible to use ordinary dice to generate random passwords.
Random password programs often have the ability to ensure that the resulting password complies with a local password policy; for instance, by always producing a mix of letters, numbers and special characters.




Danny Wind made me aware of this beautiful website

barnsten


From
€1.699,00


### Delphi 11 Alexandria

A powerful RAD environment for quickly developing high-performance native cross-platform applications using powerful visual design tools and integrated toolchains that independent developers and enterprise development teams love.

Shop Delphi

From
€1.699,00


### C++Builder 11 Alexandria

C++Builder is a complete RAD environment, loved by developers, with an integrated toolchain for modern C++ to help quickly build high-performance native Windows apps 10x faster than competing solutions.

Shop C++ Builder

From
€2.999,00

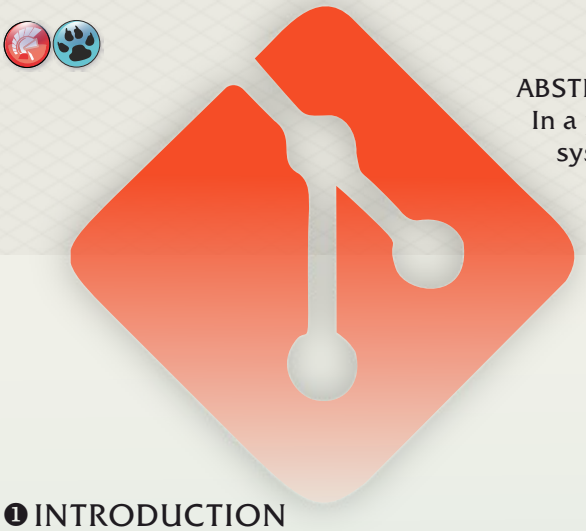

### RAD Studio 11 Alexandria

RAD Studio is the ultimate RAD environment loved by developers for quickly building high-performance native cross-platform applications in Modern C++ and Delphi using powerful visual design tools and integrated toolchains.

Shop RAD Studio

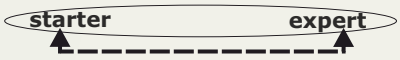

# RAD Studio 11 Alexandria

Phone number: +31 23 542 22 27 https://www.barnsten.com/



ABSTRACT

In a previous article, we introduced the distributed version system Git, and we showed how to use it to fetch and update sources from a remote repository. In this article, we'll show you how to save changes you made and how to send these changes to the remote repository.




## ❶ INTRODUCTION

Using **Git** just to download some files from a project that you want to use is somewhat overkill: The whole point of using git is to be able to make changes, and to send those changes back to the remote repository from where you got the sources.

If you do not plan to contribute to a project, it is much easier to just download a zip with the sources. Both **Gitlab** and **Github** automatically make an **URL** available to download the sources of a branch of a project. For **Gitlab,** this **URL** is:

```
https://gitlab.com/PROJECT/-/archive/ BRANCH/
REPONAME-BRANCH.zip
```

Here you must replace `PROJECT` with the **URL** of the project repository, `BRANCH` with the branch name and `REPONAME` with the name of the repository. For example, to download a zip with the latest sources of **FPC,** this is the **URL** to use:

```
https://gitlab.com/freepascal.org/
fpc/source/-/archive/main/
source-main.zip
```

The extension can be changed to other formats such as .tar, .tar.gz or tar.bz2, if you prefer one of these formats: Gitlab and Github will determine the archive format to use from the extension you added.

A similar mechanism exists for github:
`https://github.com/PROJECT/archive/refs/heads/BRANCH.zip`

So for the Free Pascal sources, the URL becomes:
`https://github.com/fpc/FPCSource/archive/refs/heads/main.zip`

But a source code management system such as **Git** is meant to be able to change the sources, and send the changes back to the originating repository:
If you have downloaded the sources of some project and you wish to contibute some changes back to the project, 2 steps must be done:

1. Record the changes locally (called committing).
2. Send the changes back to the remote repository (called pushing)

This is different from version systems like Subversion, where a commit is immediatly sent to the remote repository. We'll take a look at both steps in more detail.

## ❷ RECORDING A CHANGE: WHAT MUST BE RECORDED ?

Before recording changes, you need to know what the changes are that need to be recorded.**Git** can tell you what files you have changed (for the files it is tracking) and what files are in your copy of the repository that it does not know about.
The Git command which tells you this, is called `status`.
On the command-line, it looks like this:

```
> git status
On branch master
Changes not staged for commit:
    (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
modified: myunit.pp

Untracked files:
    (use "git add <file>..." to include in what will be committed)
myotherunit.pp

no changes added to commit (use "git add" and/or "git commit -a")
```

How to interpret this?

- The list of files after `Changes not staged for commit` is the list of files that are changed locally.
- The list of files after `Untracked files` is a list of files that exist in your copy of the sources, but which **Git** is not tracking.

If you work with `TortoiseGit`, then you can see the status of the files right in the **Windows Explorer,** because the file icons will have a marker if they are changed:

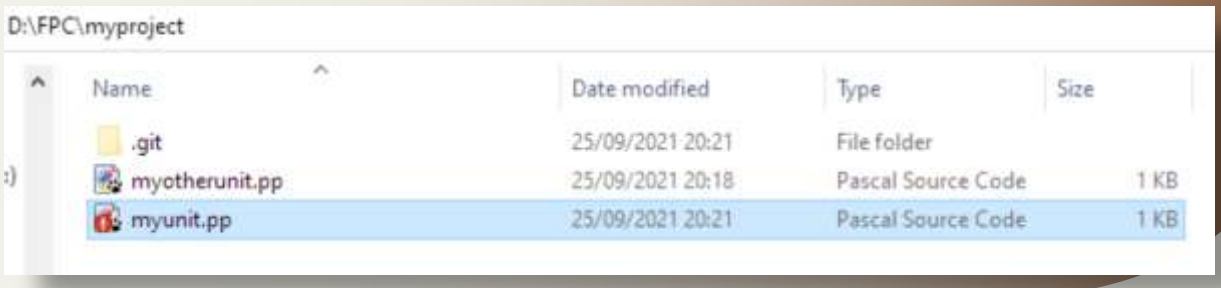



Figure 1: Icons have a marker if they are changed.

Unversioned files have no marker. For more detail, you can use the context menu of the explorer to select `Git check for modification`. Doing so will pop up the dialog shown in figure 2 on page 3, which shows essentially the same information as the `git status` command. You can modify the view by checking or unchecking some of the options in the lower-left corner.

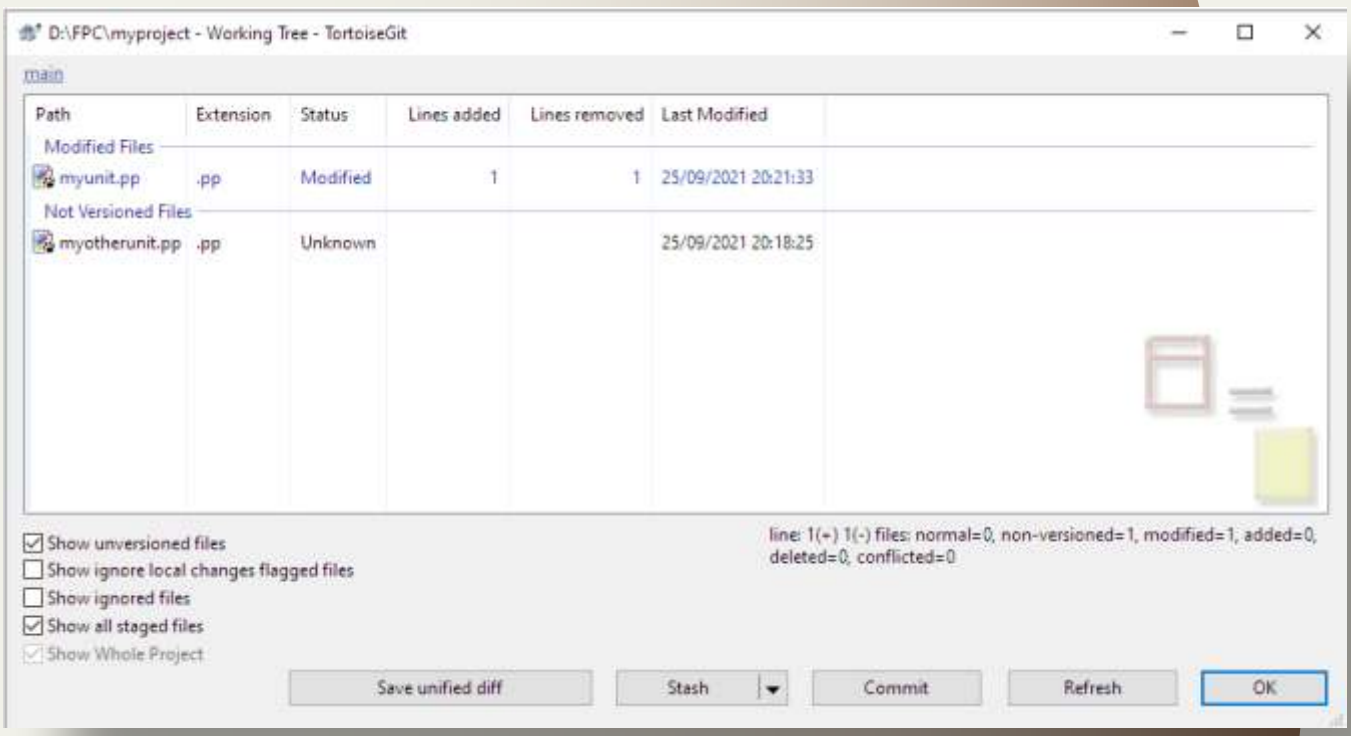



Figure 2: TortoiseGit modifications dialog

## ❸ RECORDING A CHANGE: ADDING A NEW FILE

The simplest case of a change you can make is simply adding a new file to the repository. **Git** needs to be told that you wish to start tracking changes to a file. So, the file must be added the repository.
Adding a file is done – not surprisingly – with the add command, for example:

```
git add myotherunit.pp
```

After this, **Git** knows you want to track changes to the file myotherunit.pp.
You can check this using the status command again:

```
> git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: myotherunit.pp
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: myunit.pp
```

The list of files after `Changes to be committed` is the change that **Git** will record when you actually commit the change in the next step. This list of changes is often called the `INDEX`.
The process of adding a file can be done for as many files as you want: as long as you do not commit the changes, the change is not complete, i.e. it is not yet recorded.
To do this in **TortoiseGit,** the context menu of the file explorer can be used:

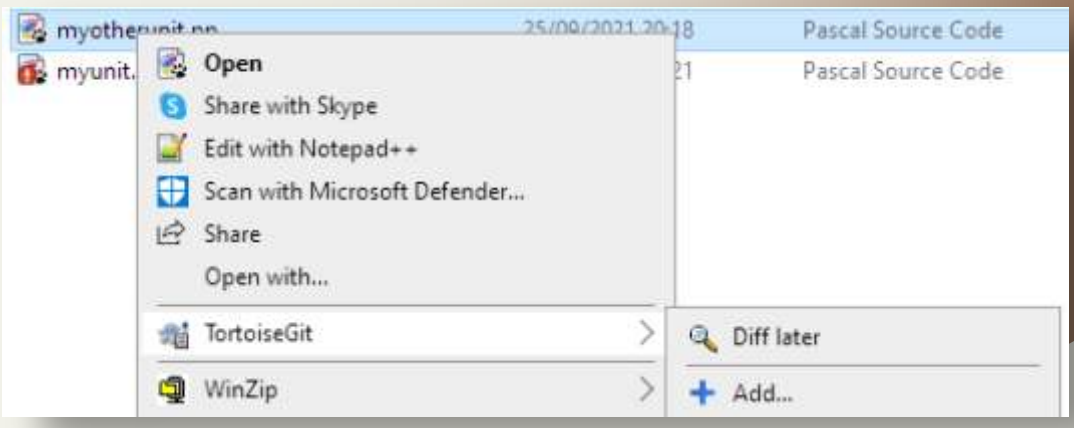



Figure 3: In TortoiseGit the context menu of the file explorer can be used:

Or the context menu of the status dialog:

In fact, in any TortoiseGit dialog that shows a list of files, the context menu can be used to do some common Git operations. Once you chose Add, TortoiseGit will confirm that the file has been added, see figure 2 on page 5. You can also see it in the explorer window:

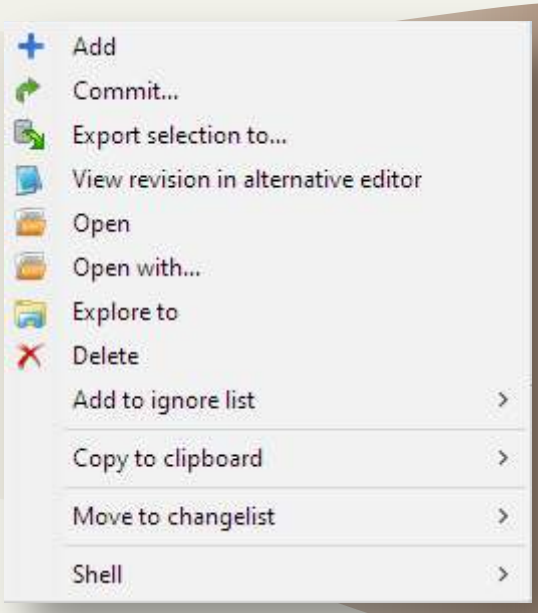



Figure 4: The Context Menu

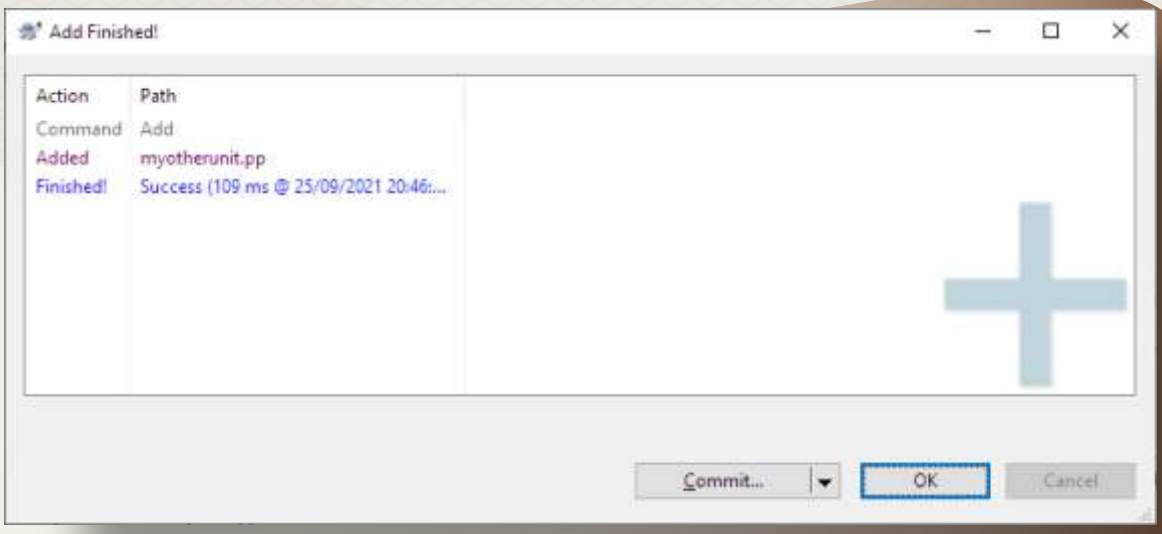



Figure 5: TortoiseGit file added confirmation dialog

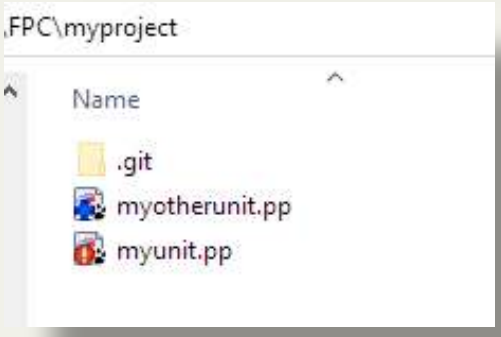



Figure 6: The icons change

You can also see it in the explorer window:

## ❹ RECORDING A CHANGE: ADDING A CHANGED FILE

The `git status` command showed that a modified file (myunit.pp) was present. To record the change in this file, it must also be added to the commit.

```
git add myunit.pp
```

This tells **Git** that you wish to record the changes of the file in its current state. When you run `git status` again, you'll see the following:

```
> git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: myotherunit.pp
modified: myunit.pp
```

In TortoiseGit, there is nothing to do. There, all actions are performed when committing.

## ❺ COMMITTING THE CHANGE

When you're done recording changed items, it is time to finalize the recording of the changes. This is done with the commit command:

```
git commit -m 'Some new hash function'
```

The -m command-line option allows you to enter a commit message. If you do not use this option, Git will automatically invoke your editor program and you can write a commit message there. It is good practice to enter a descriptive commit message: it helps developers to search for a particular change.
Git will reply with a summary of the actions it did:

```
[master 2398228] * Some new hash function
2 files changed, 23 insertions(+), 1 deletion(-)
create mode 100644 myotherunit.pp
```

After this, the status of all files will be reported as up-to-date:

```
> git status
On branch master
nothing to commit, working tree clean
```

In TortoiseGit, commiting happens in the commit dialog, which is invoked from the explorer context menu:

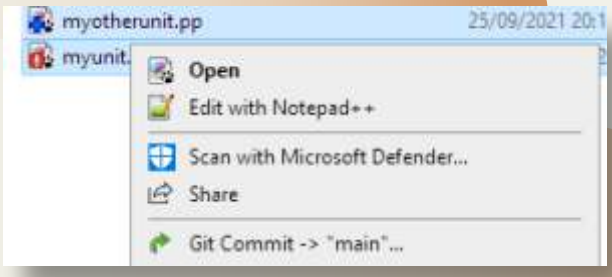



Figure 7: The Explorer context menu (Right click)

It can also be invoked from the 'Check modifications' dialog shown earlier. When selected, the dialog shown in figure 8 on page 7 pops up. In this dialog you can easily select which files need to be committed, and if need be you can still add files that were not yet added to the change to be committed: by checking the 'Show Unversioned Fles' checkbox, untracked files can also be shown.
In the dialog, you can add the comment to the commit, and by pressing the 'Commit' button, the commit will be performed. If the commit is successful, TortoiseGit will show a confirmation dialog, which shows basically the same information as you would see on the command-line. An example is shown in figure 9 on page 7

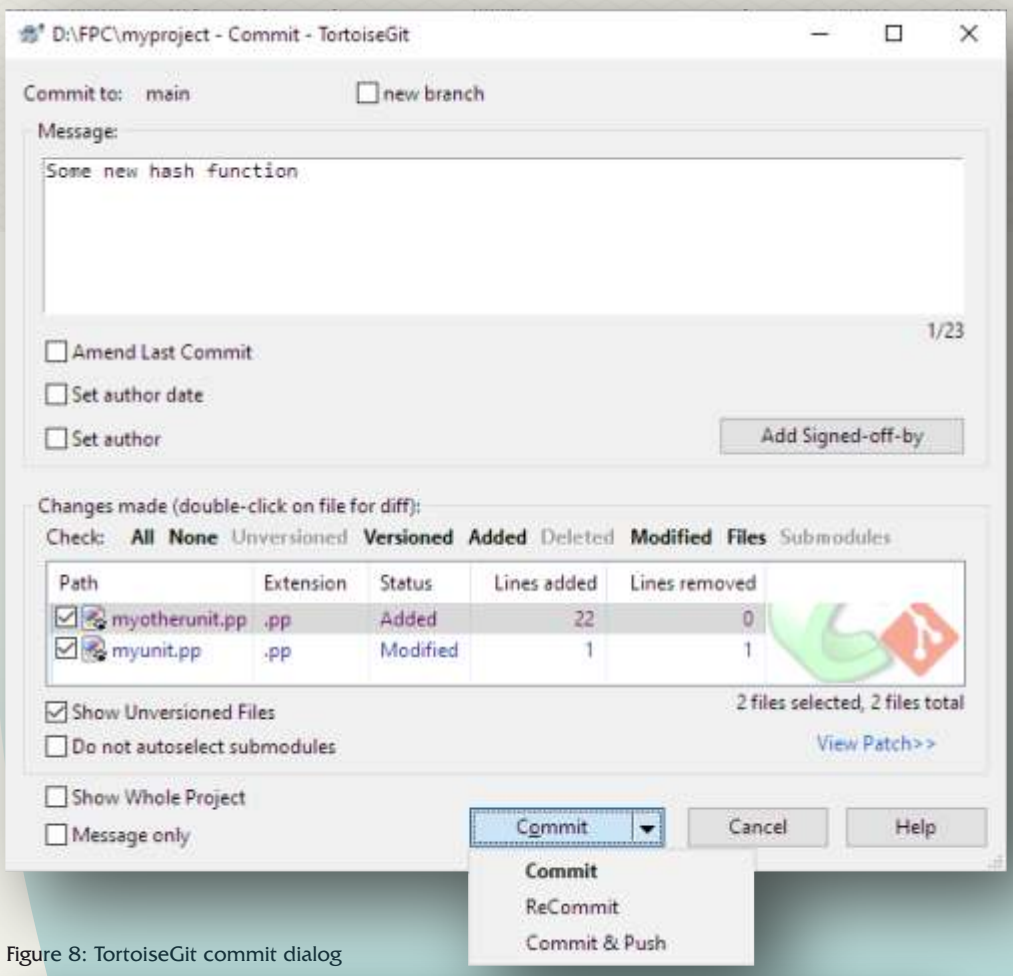



Figure 8: TortoiseGit commit dialog

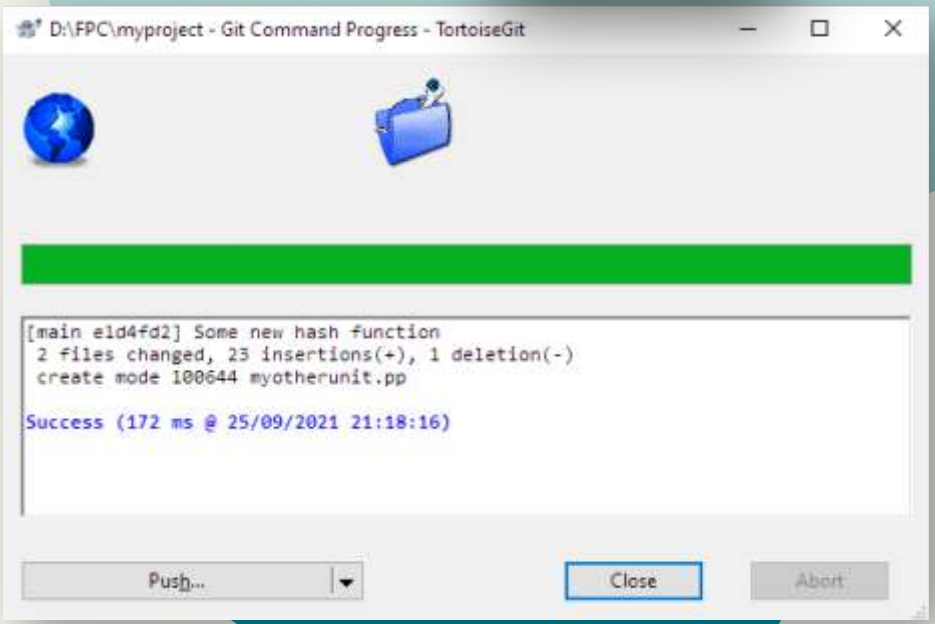



Figure 9: TortoiseGit commit result confirmation dialog

## 6 SENDING CHANGES TO A REMOTE REPOSITORY: PUSH

After the commit operation, the changes have been recorded locally: your local repository contains a new version of the files. Because Git is a distributed version system, every copy of the repository exists as a repository on its own. That means that when you commit a change, this change is effectively only recorded in your own copy of the repository. But if you cloned the local repository from a remote repository and wish to contribute your changes to the project, you will still need to send the changes which are recorded and stored in your local repository to the remote repository.

This operation is called 'pushing' your changes to the server, and the command for it is `git push`. This will compare the list of changes recorded on the remote repository with the list of changes recorded locally, and sends to the remote repository all local changes that do not yet exist on the remote repository.

How do you know where the changes will be pushed to? The git remote command will tell you this:

```
> git remote -v
origin git@gitlab.com:mvancanneyt/myarticleproject.git (fetch)
origin git@gitlab.com:mvancanneyt/myarticleproject.git (push)
```

In this output, origin is the name of the remote repository. This name is created automatically when you clone a remote repository.

Because Git is a distributed version control system, there can be multiple remote repositories: each will have its own name. In that case, the remote command will list all remote repositories:

```
> git remote -v
ondrej https://gitlab.com/onpok/fpc.git (fetch)
ondrej https://gitlab.com/onpok/fpc.git (push)
origin git@gitlab.com:freepascal.org/fpc/source.git (fetch)
origin git@gitlab.com:freepascal.org/fpc/source.git (push)
```

The pull and push URLs can differ in theory, that is why git shows 2 URLs for each remote. A second (and probably more likely) question is of course: Exactly what changes will be sent to the remote server when you push? For this you can use the git log command:

```
git log origin/main..HEAD
```

The above command lists all commits between the remote revision origin/main and the local HEAD revision: these are aliases for the last (locally known) commit on the main branch on the server and the last local (HEAD) commit.

If you have multiple remote servers configured, you simply need to replace the origin with the correct name of the remote repository.

```
git push
```

If all goes well, then Git will print some diagnostic messages, telling you what it is doing:

```
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
```

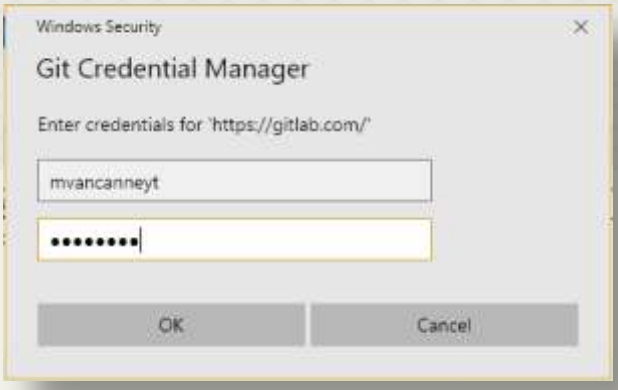



Figure 10: TortoiseGit credentials dialog

```
Delta compression using up to 8 threads
Compressing objects:
100% (7/7), done.
Writing objects:
100% (7/7), 918 bytes | 918.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
To gitlab.com:mvancanneyt/myproject.git
4fcf99a..8143d7d main -> main
```

If you wish to push only to a single remote, you can give the name of the repository:

```
git push ondrej
```

If no repository is specified, then **Git** will assume 'origin' as the name. Depending on the configuration and where you cloned from, **Git** will ask you for a username and password if the remote repository requires you to be authenticated: for Gitlab and Github this will definitely be the case. **Git** supports several transport mechanisms: HTTP(s) and ssh are among the most used.

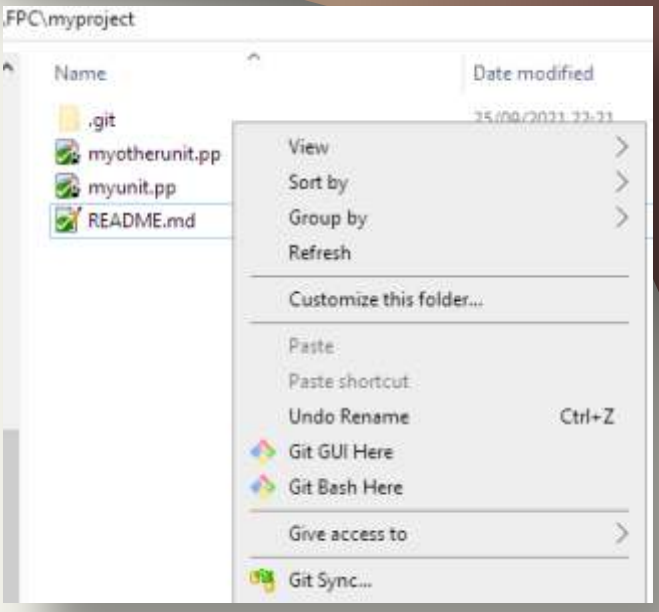



Figure 11: the 'Sync' menu in the explorer popup menu.

**ssh**

In this case, no credentials will be asked: **Git** will use the ssh keys that you configured for your SSH setup.

**https**

Here, the credentials must be entered once when you connnect to the remote repository. (*see figure 10 on page 9 for an idea of what this looks like in TortoiseGit*)
By default **Git** (**or TortoiseGit**) will then save them so they will be used the next time you need to authenticate.

To push changes to a remote repository in TortoiseGit, the 'Sync' menu item in the Explorer's context menu must be used.

When selected, the synchronization dialog pops up, it looks like figure 12 on page 11. This dialog allows you to synchronize your local repository with a remote repository: It allows you to select the repository to which you will push changes and it shows the list of local changes that will be pushed to the selected repository: this is essentially the output of the git log command mentioned earlier.

There are some other options, but we will not discuss them at this point.

For the push operation, the Push button below the list of changed files must be used. If all goes well, then **TortoiseGit** will show the result of the operation, just as it is shown when pushing on the command-line, see figure 13 on page 11.

It can happen that the push operation fails: if the remote repository has received changes from other contributors which are not yet present in your local repository, then the server will refuse to apply your changes. In that case you will first need to pull the changes from the remote repository, and when that was successful, you can attempt to push your changes again.

That is also the reason why **TortoiseGit** has a single 'synchronization' dialog and not separate dialogs for push and pull: The 2 operations often must be executed one after the other, and this is easier when the operations can be performed from a single dialog.


Figure 12: TortoiseGit synchronization dialog

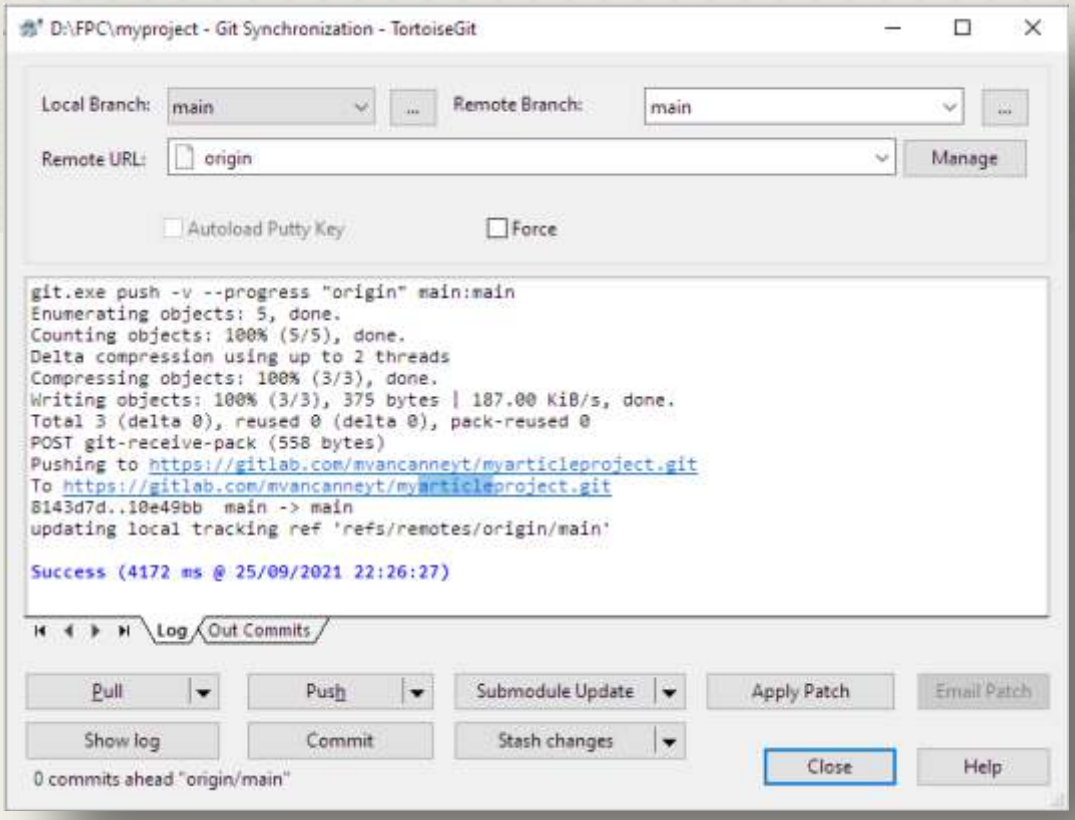



Figure 13: TortoiseGit synchronization result confirmation dialog

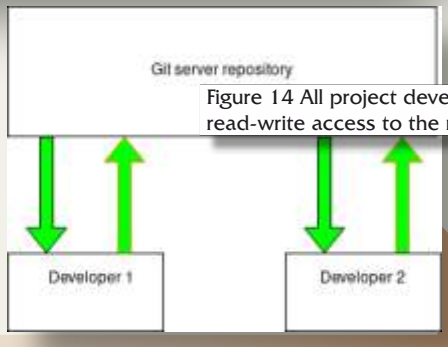



Figure 14 All project developers have read-write access to the remote repository

## 7 COLLABORATION: USING FORKS

When you clone a repository from Gitlab or Github to your local machine, chances are that you have done so for a repository that you don't have write access to: most projects do not allow arbitrary developers to push changes directly back to their hosted repository.

Instead, only a limited amount of people will have write access to the repository on **Gitlab** or Github, a situation depicted in figure 14 on page 11. So if you don't have write access to the remote repository, how can you contribute your changes to the project?

The TortoiseGit 'synchronization' dialog (*figure 12 on page 10*) shows one possibility: The 'Email patch' button can be used to create a patch file and mail it to one of the developers of the project. The project developer can then use the 'Apply patch' button to apply your patch to his or her local repository and push it to the remote repository: he will presumably have the necessary authorizations to perform the push. This of course implies that you must know the email address of one of the developers.

However, the creators of **Gitlab** and **Github** have created an easier way to send your changes back to the project repository: Forks. Git is a distributed version system: this means that there can exist many copies of a repository, and they can all be kept in sync in a more or less automated manner. What sites as Gitlab and Github do is to allow you to create a copy of a repository on their servers: this copy is called a fork.

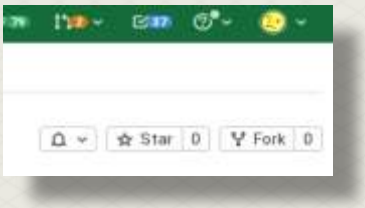



Figure 15: Gitlab fork buttons

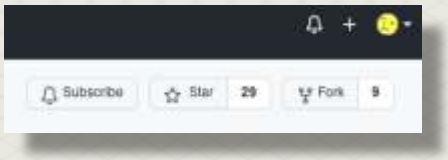



Figure :16 Github fork button

A similar button exists in github's project page:

In Gitlab, this is done simply by clicking the 'Fork' button in the project page of the project you wish to fork: In order to create a fork, you must have an account on **Gitlab** or **GITHUB.** The copy will be registered under your account, and the developers will see that you have created a fork. In difference with the original project repository, you will have full control over the fork. The situation and possibilities are depicted in figure17 on page 12.

- You can clone your copy locally, and write from your local copy to your forked repository on the server (the 2 green vertical arrows at the right).
- You can also set up your fork to automatically pull changes from the original project repository, so you will always get the latest changes from the original repository (*the dark blue arrow at the top*).
- You can even register 2 remote repositories in your local copy: your fork and the original repository. If you do this, you can manually keep your forked repository up to date: you pull changes from the original project repository (*the light-blue arrow pointing down-right*), and push them to your fork (*green up arrow*).

So how can you contribute changes in this scenario? This is depicted using the orange arrow in figure 18 on page 12. The way to send back your changes is called a merge request. When you push changes to your fork, the Gitlab or Github servers can be configured to automatically send you an URL which you can use to create a merge request. This URL will take you to their website, where you can fill in the details of the merge request.
For **Gitlab,** the merge request page looks like figure 18 on page 12 and 13 of this particle.

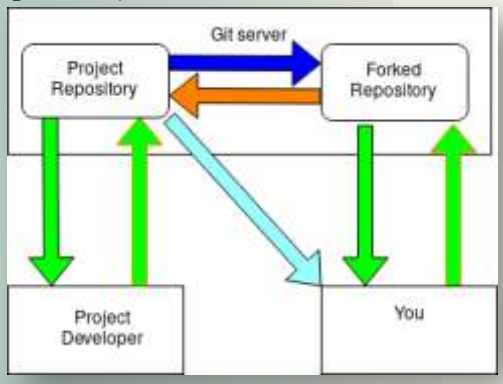
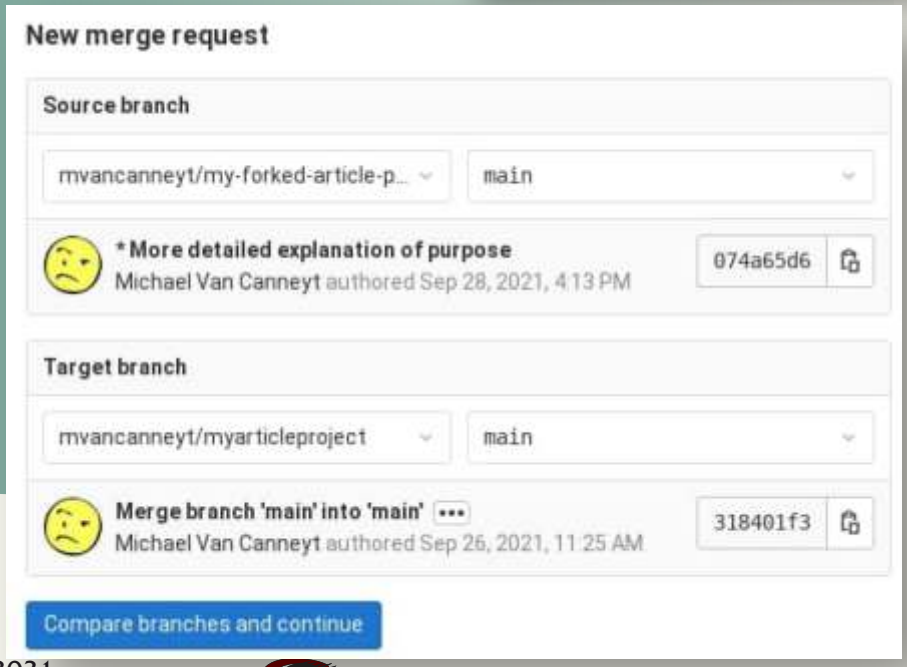



Figure 17: Flow when using a fork

Figure 18: Starting a merge request



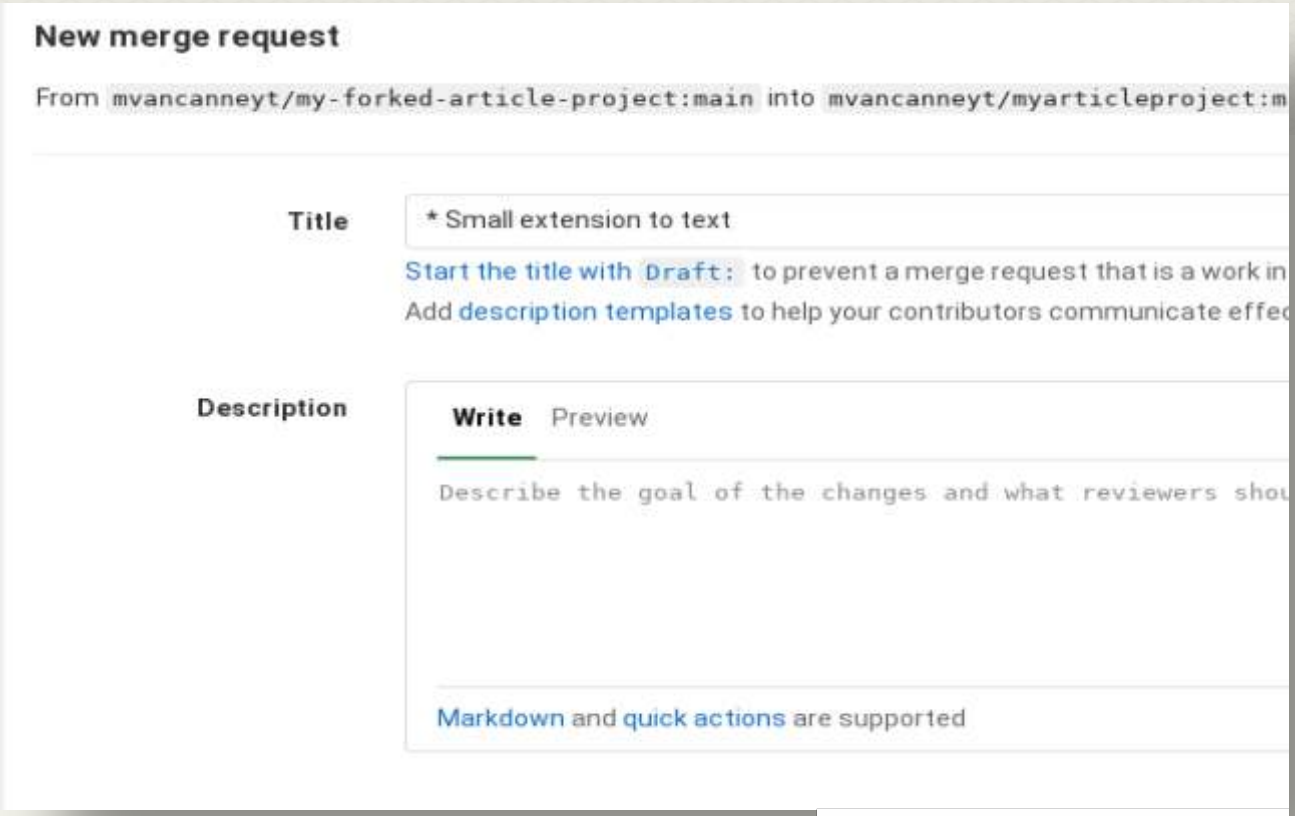



Figure 19: Provide info about your request

You select the source repository (*your fork*) and branch (*main, or the branch where you committed your change*) and target repository (*this is normally the project repository*) and branch where the change is supposed to go.

When you continue, a dialogue appears (*figure 19 on page 13)* where you can give the developers of the project an explanation of what your patch does, what problem it solves, or why you think it is a necessary or useful change: in general, provide any info to help the developers decide whether or not they want to incorporate your change into the project.

It is worth noting that although you will be creating the merge request starting in your forked copy of the project repository, the actual merge request will end up in the original project. You will not see it in the list of merge requests of your project.

For **Gitlab,** all this can be done on the command-line as well, but the process is rather involved and requires extensive command-line skills.

A merge request is just what the name says: a request to merge a change.
This has several consequences:

- The change is not automatically merged into the repository of the developers.
- Instead, it is recorded in the **Gitlab** or **Github** platform.
- The developers are notified of your request:
  they receive an email with the details of your request.
- The request is shown to them when they go on the website
  (*see figure 20 on page 13 for how this looks for a developer of a project on gitlab*).
- They can examine your changes using the website,
  or check out your version of the files.
- The developers can create comments on your merge request, add comments to details of the diff (*figure 21 on page 15 shows what a developer sees from your merge request*).

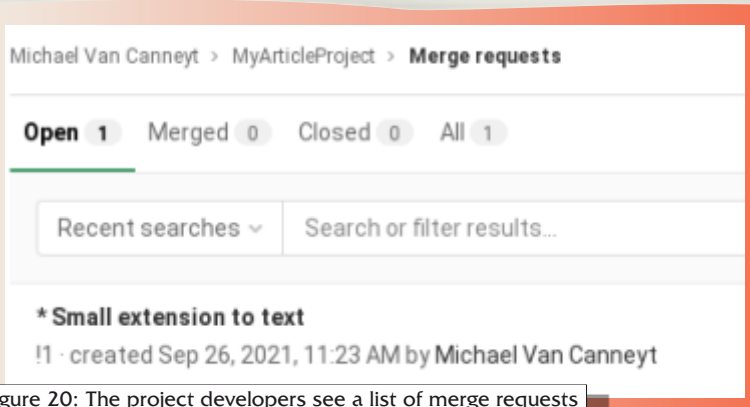



Figure 20: The project developers see a list of merge requests

- The developers can request additional changes: if you make additional changes in the same branch and push them to your forked repository they will automatically be added to the merge request.
- There can be an approval process.
- Automated tests can be run.

There is a large list of possibilities: a whole ecosystem of options exist to help the project developers to examine and process your request.
If all went well, the project developers decide to accept your merge request.
The merge request is then approved (*although this is optional*) and the change is actually merged into their repository: this can be done in 2 ways:

- Using the button in the website. In that case, the request looks like figure 22 on page 15.
- By applying the changes on a local repository and pushing them to the server.
  Depending on the way this is done, the merge request will also be closed automatically.

The setup of platforms like **Gitlab** and **Github** offer a lot of configuration to automatically handle merge requests. If CI (Continuous Integration) was set up, then they can for example configure the server to automatically run a test suite on the sources of every merge request and even allow your merge request to be merged automatically if the test suite runs without fail. The details of this depend of course on the service level of your project account and the hosting platform you are using.

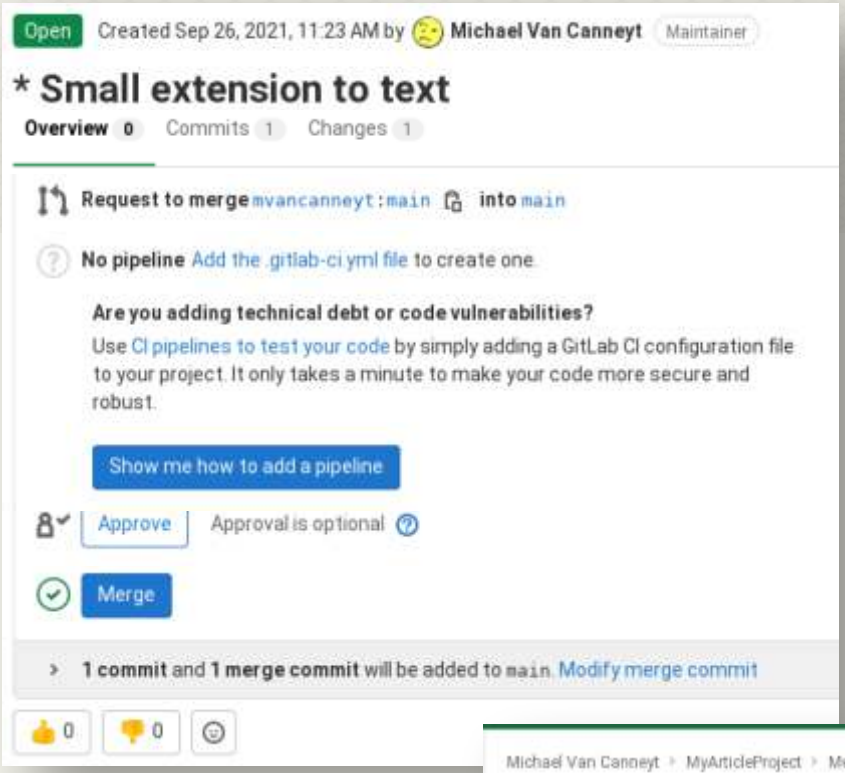



Figure 21:
Details about the merge request

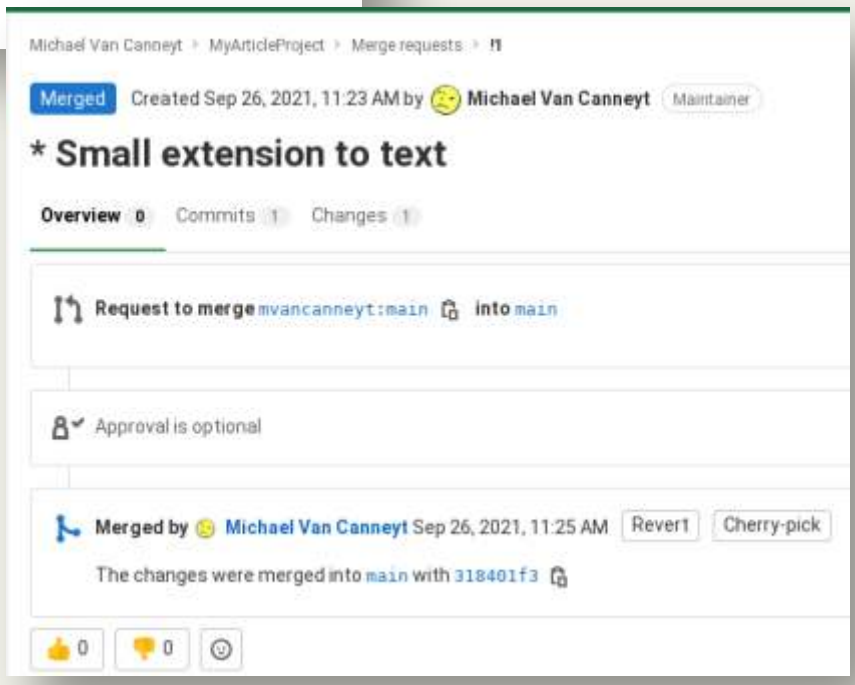



Figure 22: The merge request was accepted and merged

## ⑧ CONCLUSION

In this article, we have shown how to communicate changes to a remote repository, and how this can even be done when you do not have write access to the project's repository: Forks are the feature of platforms such as Gitlab and Github which make Git such a succesful collaboration tool. But we have not yet covered all of Git - not by a long shot.

**In the next Git article, we'll cover the use of splitting large changes into smaller commits and the use of branches in Git, as it is also a major topic when co-operating on a project managed by Git.**

# THE DELPHI COMPANY

-est 1998-

OS X Android iOs Windows

Four Platforms
One develop environment
One Expertise

Vier platforms
Eén ontwikkelomgeving
Eén expertise

**DELPHI**

www.delphicompany.nl
info@delphicompany.nl

Fast Reports
Reporting must be Fast!

# Uploading objects to FastReport VCL using the http and https protocols

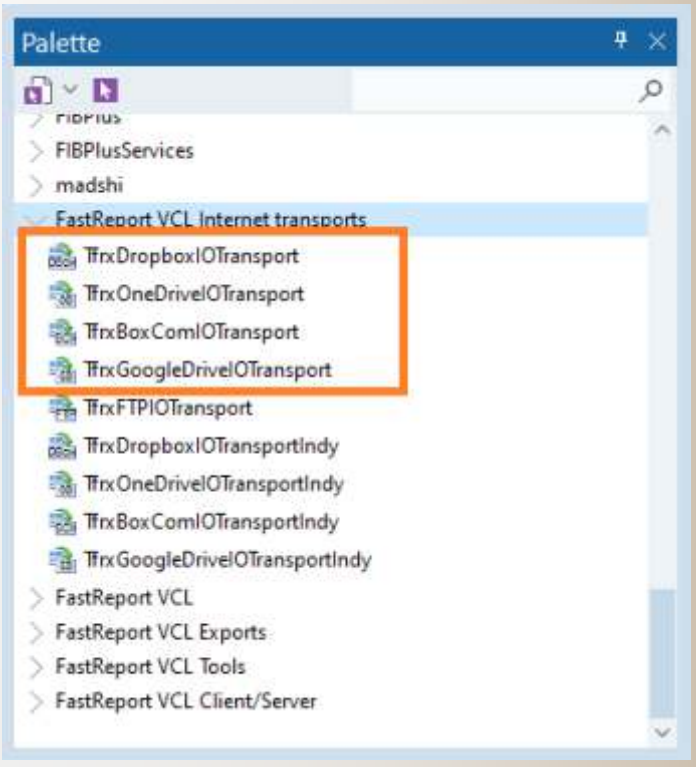


One of the key tasks of a report generator is data retrieval. Most often, databases are used to store and retrieve data. But what if it is remote cloud storage, which is used to store certain data? How to upload a company logo via http to a report to ensure that it is always up-to-date?

To solve these problems in FastReport VCL 2021.3 we have added support for protocols and the function of loading data through protocols into objects such as: Maps, Picture, Text, PDF object, HTML object. Some objects have a new **DataLink** property that includes additional properties for handling links. Such links always start with the schema name followed by a separator. Example <schema>: [//path]. In most cases, the scheme is protocols such as **http** and **https.**

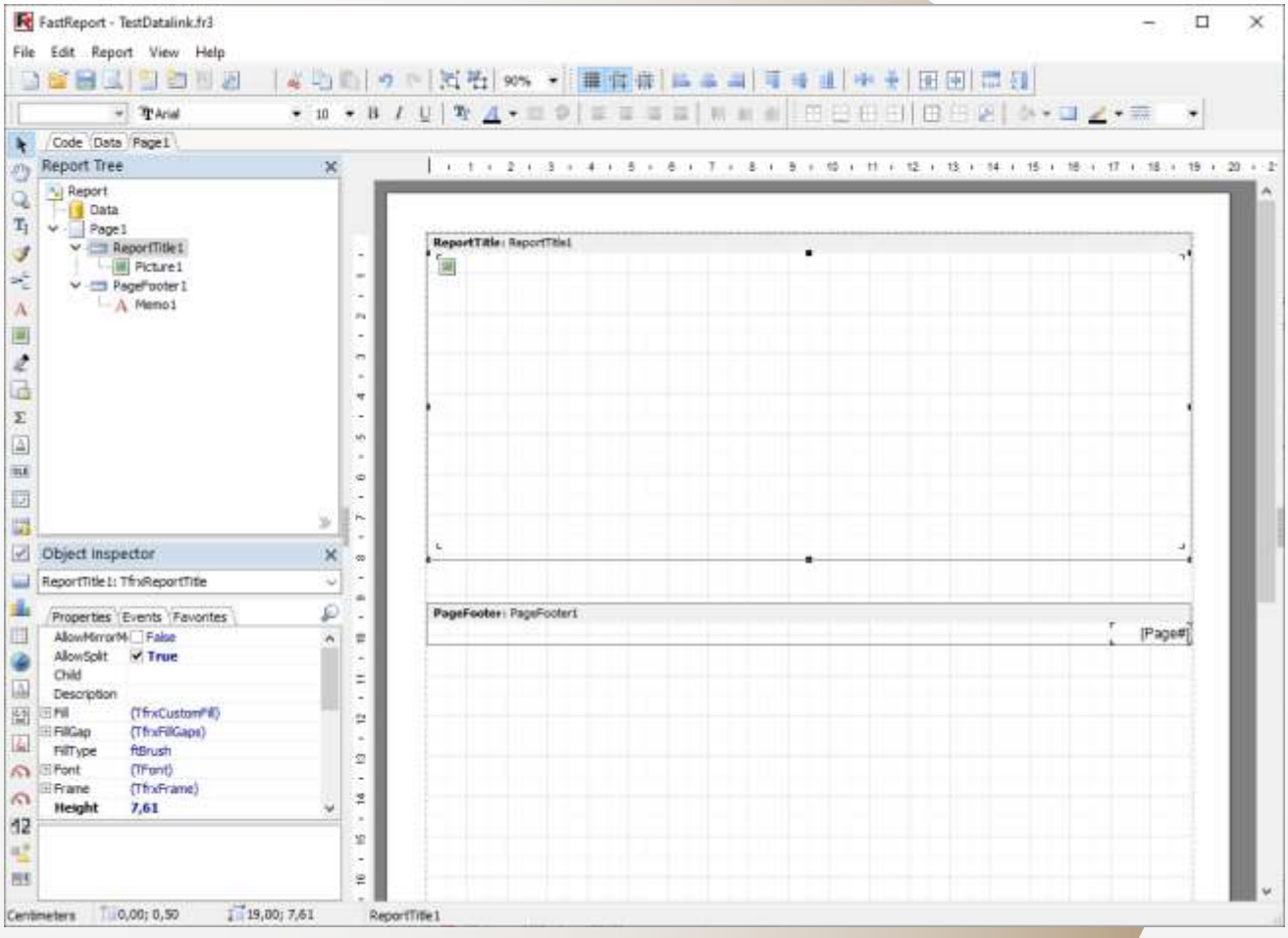
## Data loading via links in FastReport VCL

FastReport VCL is based on a modular architecture, the functionality of http and https is no exception, and it is included in a separate package along with Internet transports. Therefore, for links to work via the http and https protocols, you need to connect the frxTransportHTTP module in the uses section of the application or add one of the Internet transports to the application form.

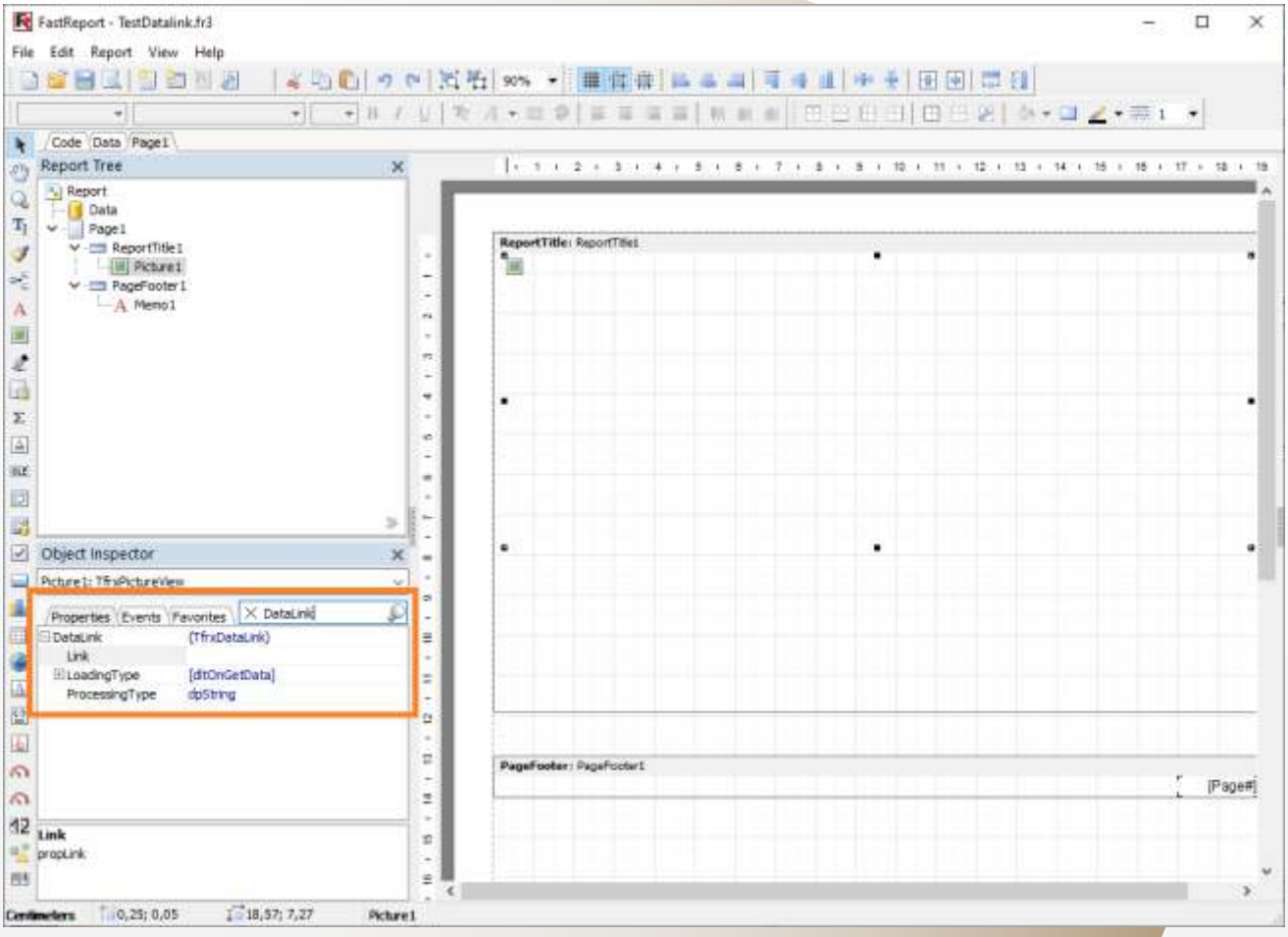



*Important! The following Open SSL libraries are required for the https protocol to work: ssleay32.dll and libeay32.dll. You can find them in the directory with the main demo application.*

Let's start the report designer and create a new report. Then add the "Report Title" band on the report page and a picture object to it. The report should look as follows:




Let's select the "Picture" object and go to the object inspector. Find the DataLink property in the object inspector. For a quick search, you can use the filter built into the object inspector by entering the name of the desired property.

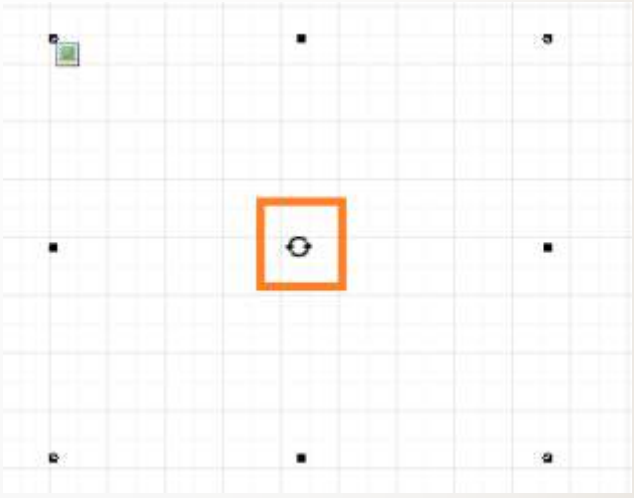



Let's look at the meaning of the DataLink children properties:

Link – a link with a schema (protocol), in the format <schema>: [//path]. The data will be loaded from the specified link.

LoadingType – loading type. It indicates when the data will be loaded.
It can have the following values:

[] – Empty. The data is loaded in the template editing mode, when the user clicks on the load editor. It is used if you need to retrieve data only once and store it in the template.



- [dltOnGetData] – By default. The data is loaded into the object at the time of building the report. In this case, the object data is not saved in the template. The object receives new data via link each time the report is generated.

- [dltOnPreview] – Data is loaded during report preview, export or printing. In this case, the generated report stores a link to the data and loads it every time the user is loading the generated report. It can be used to get up-to-date data in the generated report without rebuilding it.
- [dltOnGetData, dltOnPreview] – Hybrid mode. When building a report, the object receives data via link and saves it in the generated report. The link to the data is also saved. When loading the generated report with this object, the object will try to load data in the same way as with [dltOnPreview], but the data obtained during the report generation will be shown if data loading was unsuccessful.
- ProcessingType – a value that determines how the link will be processed during report generation: dpString – as a regular string, dpExpression – as an expression of the report script. Let's set the DataLink.Link property of the "Picture" object with a link to a picture from the website, for example. Let's execute the report.

Let's set the DataLink.Link property of the "Picture" object with a link to a picture from the website, for example. Let's execute the report.

Fast Reports
Reporting must be Fast!

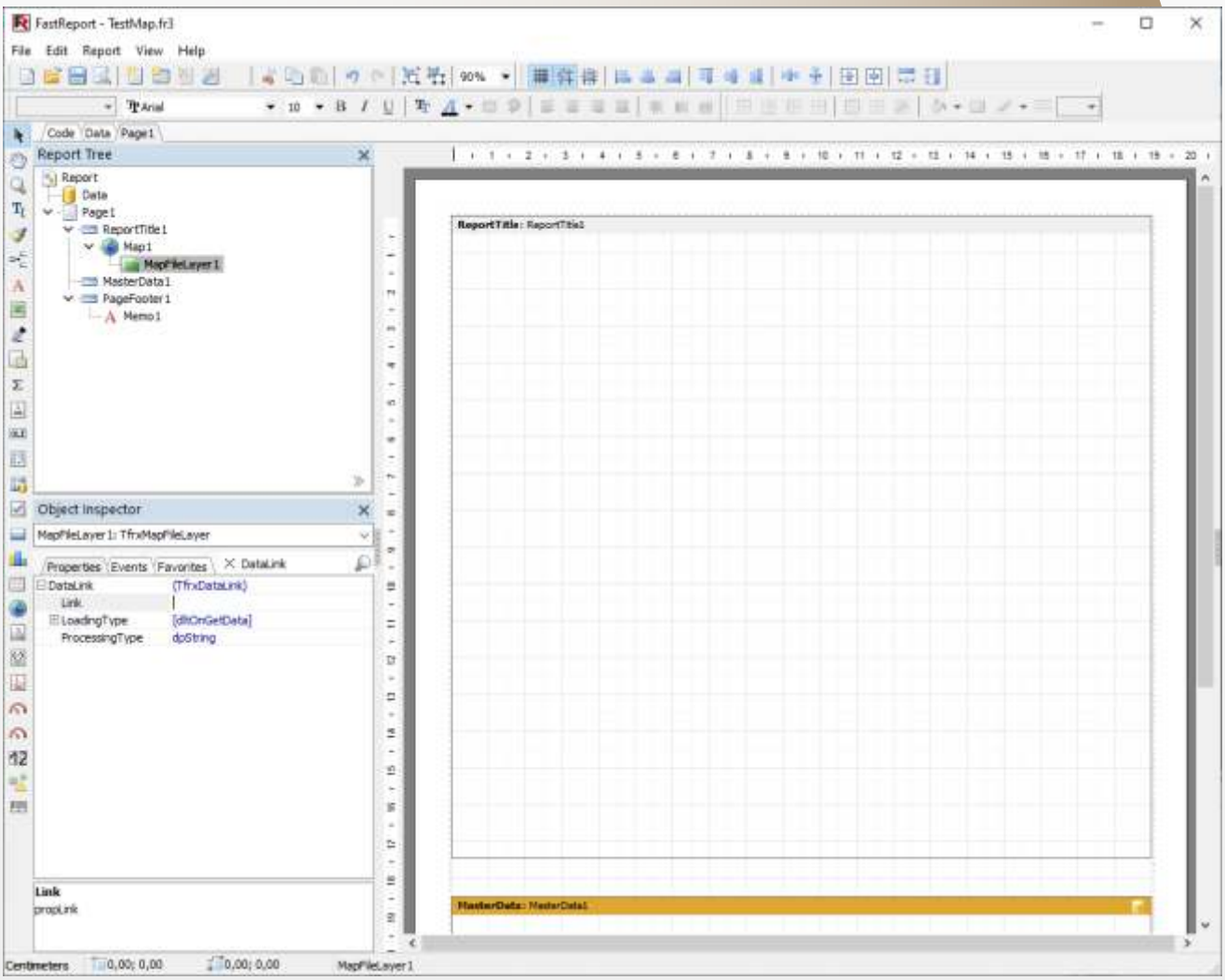
# Uploading objects to FastReport using the http and https protocols




Similarly, you can load other objects, such as: Text, Picture, Maps, HTMLView, PDFView.

The links can be used to access the Web API to get data, for example OSM maps.

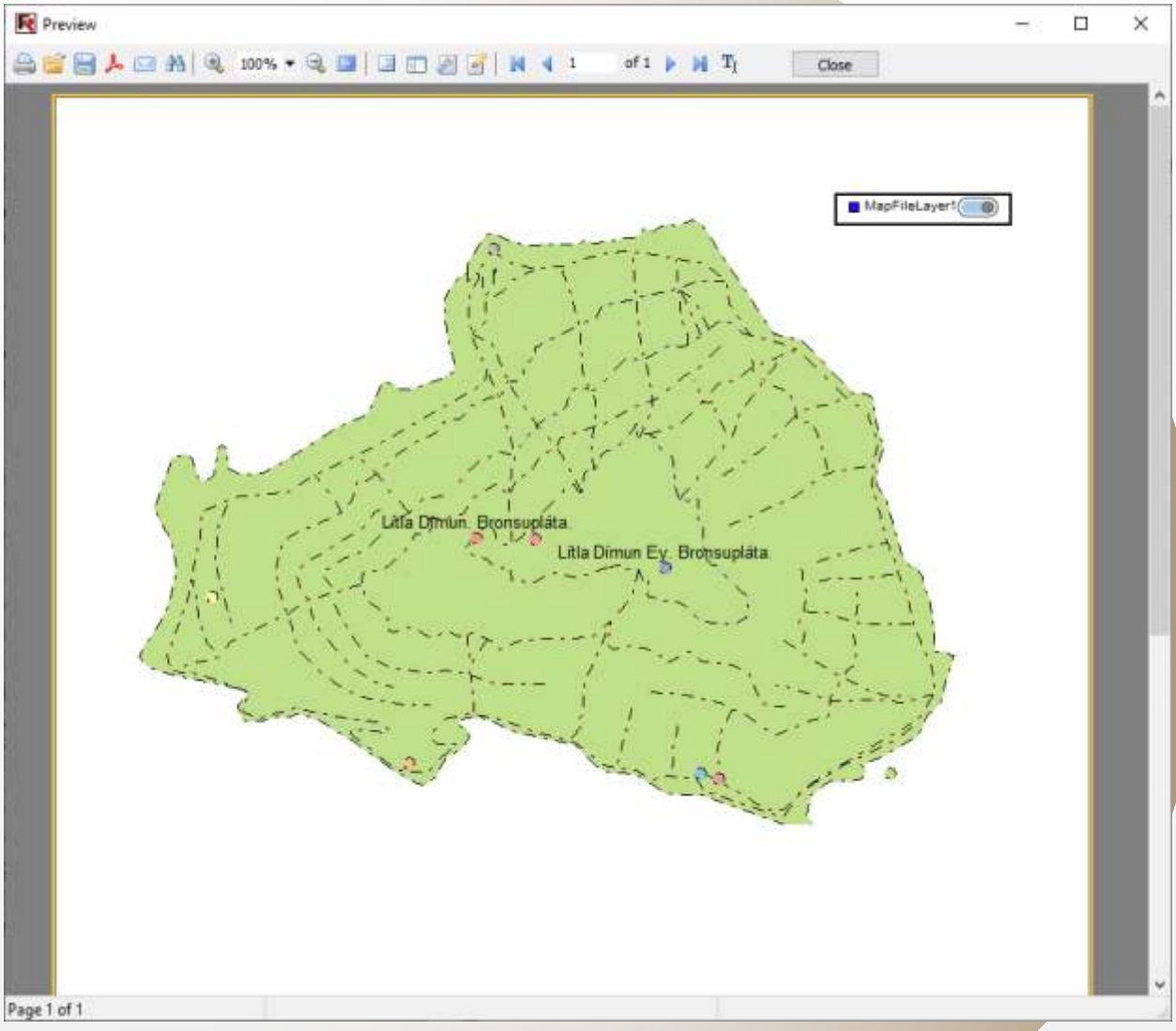
Let's create a new report with the "Report Title" band and a Map object with an empty layer (You can see how to create the Map object in our documentation).



Now let's select the map layer MapFileLayer1 and go to the DataLink.Link property in the object inspector. You can use the OpenStreetMap API to load OSM maps. Let's try to display the selected area in FastReport VCL. The API link will look as follows:

```
https://www.openstreetmap.org/api/0.6/map?bbox=-6.7234%2C61.6283%2C-6.697%2C61.6379.
```

Insert it into the DataLink.Link property. Let's generate the report.



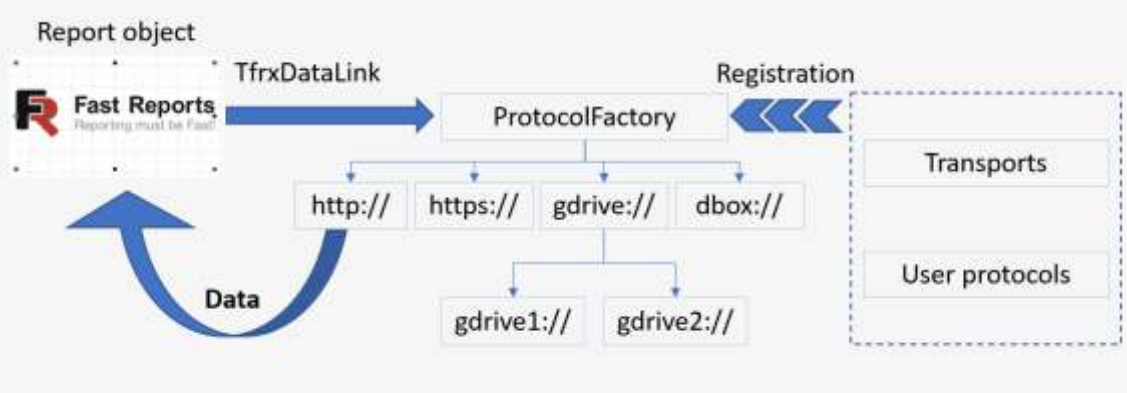

The data from the server has been loaded into the map object automatically.

## More than hyperlinks

In FastReport VCL, all schemas and protocols of the DataLink property are processed through a factory. This allows developers to expand the functionality of the application by adding new schemes and protocols without any changes in the FastReport VCL source codes. In addition, FastReport VCL automatically registers internet transports in the factory as new protocols. This enables to access private cloud storage and load data from them into a report, without transferring or storing any authorization data within the report. You can see the scheme of interaction of the DataLink property with the protocol factory below.

# Uploading objects to FastReport using the http and https protocols

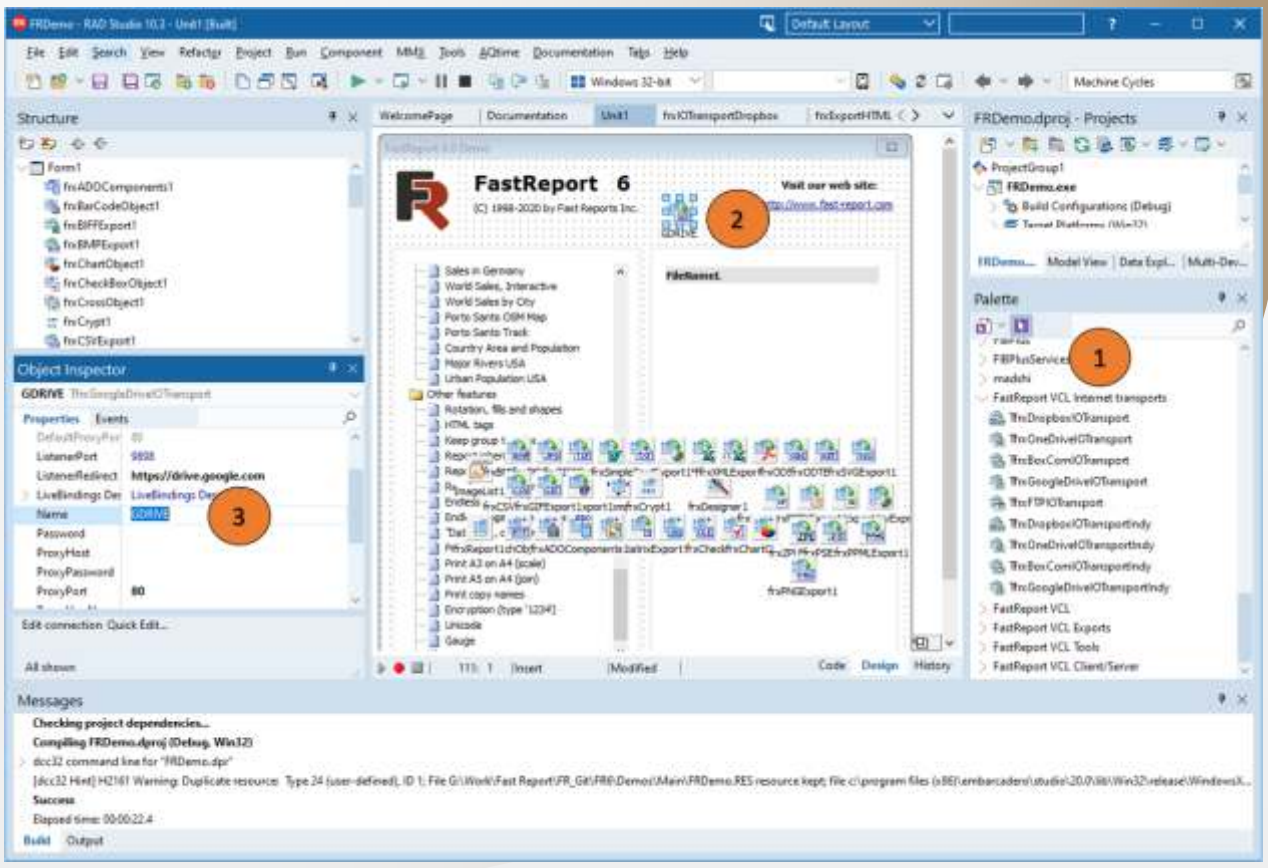




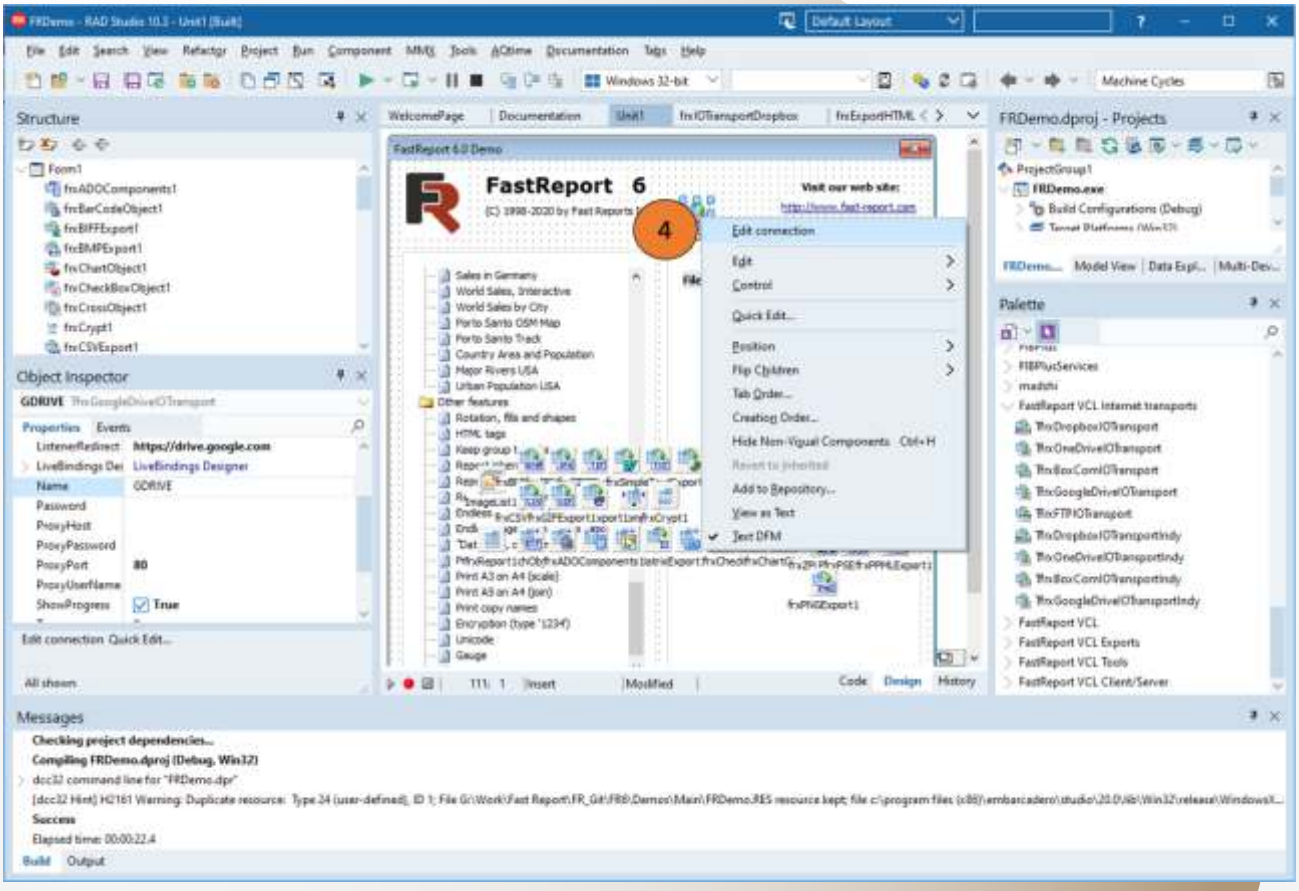

How to add internet transport and use links to a private Google Drive?

This can be done in just a few steps:

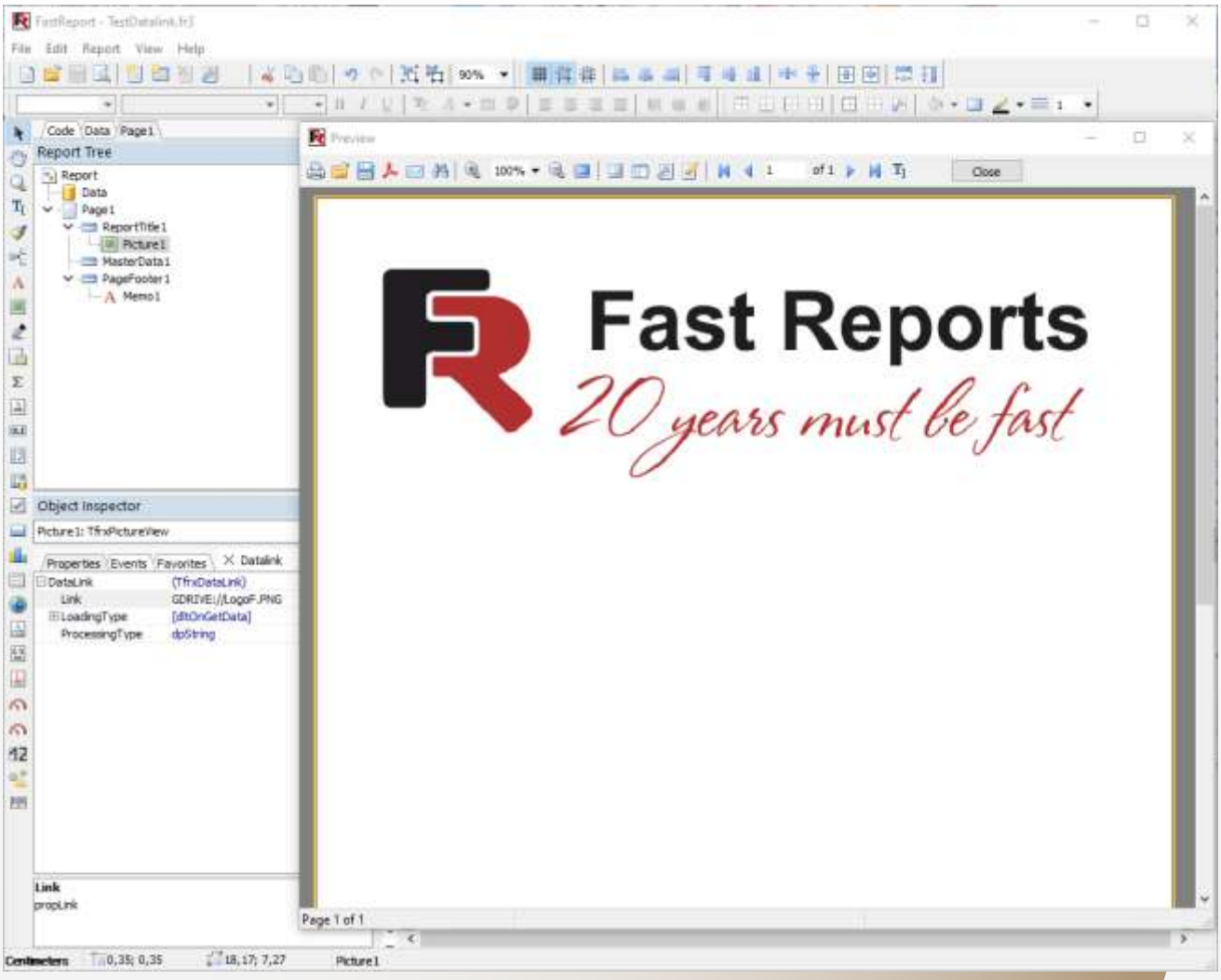
1. Open the component palette with Internet transports.
2. Add TfrxGoogleDriveIOTransport to the application form.
3. Assign a name to the component. The name will be used as the schema protocol for the access link. In the example, we named it “GDRIVE”.




Edit connection to Google Drive through the “Edit connection” menu (see another article for more details).




5. Run the report designer and create a report by following the steps as at the beginning of the article when you created a report with a picture.
6. You need to specify the link to the file in Google Dive in the DataLink.Link property, but instead of the scheme (protocol) specify the name of the transport. In our case, it is “GDRIVE”. The link will look like this: GDRIVE://LogoF.PNG. Then you can execute the report.




*Important! If the data is not loaded, make sure that you've connected to the cloud storage properly and the authorization token was saved when connecting.*

Similarly, you can connect to other Internet transports. This is one of the few powerful tools that were introduced in FastReport VCL 2021.3 and which you can use to improve your projects.

Den Zubov

VCL Development

Fast Reports Team: Den Zubov - VCL Development at Fast Reports

# RECOVER ICONS ON YOUR WINDOWS DESKTOP

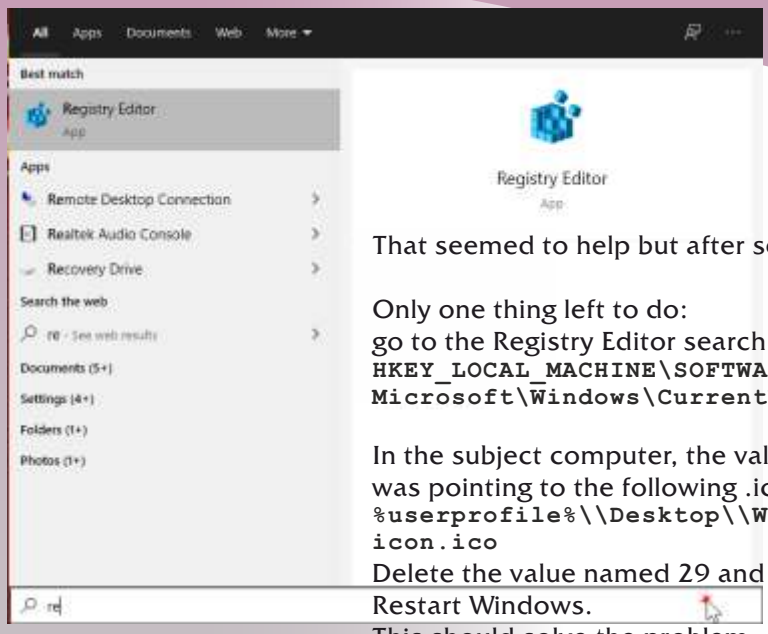
BY DETLEF OVERBEEK

Some weeks ago I had an ugly problem, I could still work of course but it was rather annoying.

I looked at several solutions that came up from the web and some official from **Microsoft** it self.
Nothing worked.

After restarting the problem reappeared,
So I tried first of all the simplest solution: remove the IconCache

```
c:\Users\yourusername\AppData\Local\IconCache.db
```




That seemed to help but after several restarts It reappeared.

Only one thing left to do:
go to the Registry Editor search for this address:

```
HKEY_LOCAL_MACHINE\SOFTWARE\
Microsoft\Windows\CurrentVersion\Explorer\Shell Icons
```

In the subject computer, the value named 29 in the above registry location was pointing to the following .ico file:

```
%userprofile%\\Desktop\\WinBubble16\\WinBubble16\\icons\\no
icon.ico
```

Delete the value named 29 and the registry should look like Figure 3.
Restart Windows.
This should solve the problem.

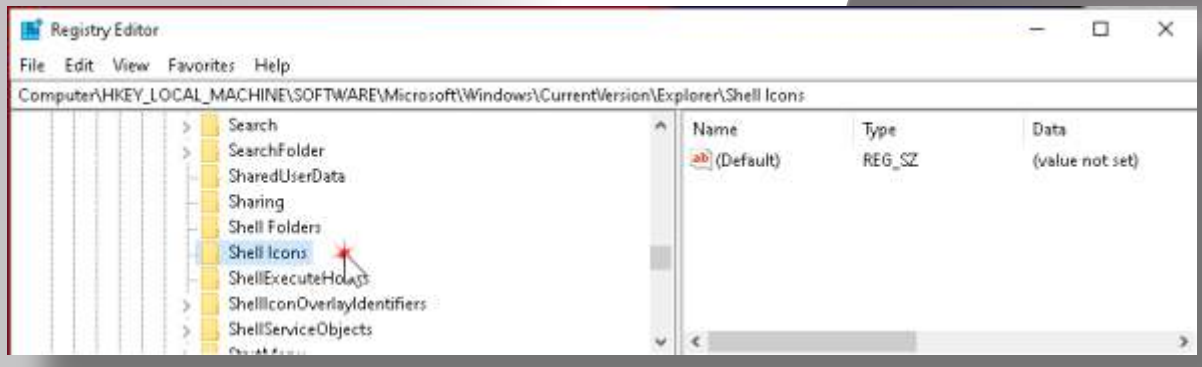



Figure 1; The splash screen

## AN OVERVIEW

A new version of Delphi has
arrived.
This time it wasn't because of the annual changes.
Embarcadero went from Delphi 10.4.2 to Delphi 11 (Alexandria).
The reason for this is that they want to follow the Microsoft announcement for the next windows version: Win11.
Of course we had all heard of that. In this article we will create an overview of the new items that are special to this new version.

Community Edition? Probably 10.4?

I have not yet started testing it under the new windows version because I need more time to test that environment. The next version of Win11 will be released on October 5, 2021. Since I have a complete new computer which has a CPU named AMD Ryzen 9 5950X 16 -Core Processor, I was surprised it didn't pass the test.

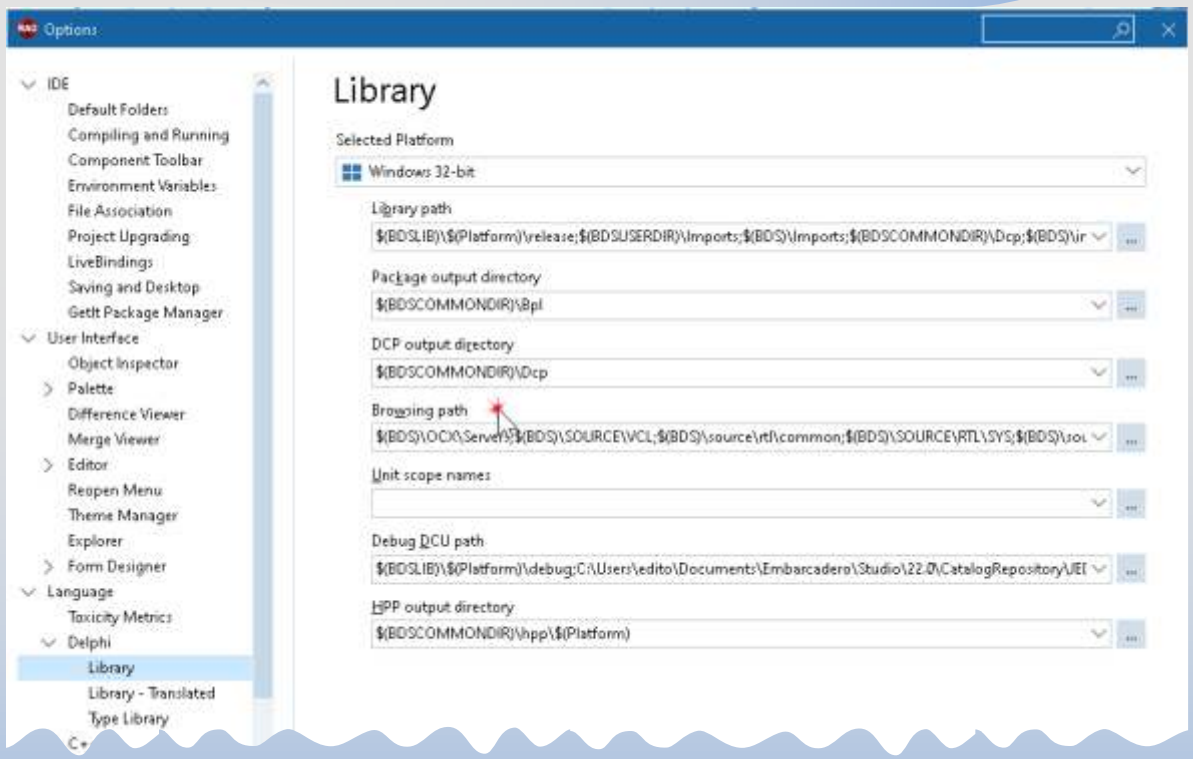
If they go for that they wont have a loft of new users.
Ill have to find out what the problems are that I can update. I will do so in a next issue 99/100 anniversary edition. So because of that we will not be able to check if all the new features will fit With Delphi Alexandria.

## COMPONENT SEARCH PATHS


So because of that we will not be able to check if all the new features will fit With Delphi Alexandria. In the rest of the article items that are interesting and important will be explained and shown.

The startup window shows after a while quite a number of components:

About installing these components needs to be explained some new issues



## COMPONENT SEARCH PATHS

It are small but confusing things. Once you try to install there are two things to be handled

❶ The settings for the IDE to find the path to your new component. It has always made me wonder why these settings are not in a section you would expect them to be? (Figure XX) The browsing path is to be found under Language. Can anyone explain that to me? I had hoped they would have altered that.

❷ You also need to arrange some settings for the project in which you want this component to be available. Go to project and choose at the bottom Options (*Figure 3*) . A new window will appear after a while. (*Figure* 4).
Under the section Delphi Compiler you need to go to the Search Path. Click that and you will find a new line. You need this. (*Figure 5 on the next page*).
Because in earlier version you could simply alter the text of that line and add to it.
The much better way they created is You cant do that any more but need to expand the lines below search path (*Figure 6*) and you will find some new lines where you can update or alter your settings.
You can clean up in detail (*which wasn't that evident in older versions*), delete parts or whole paths.

Good idea, because it happened often that your settings were not persevered.
This is one of the many items that are improved.

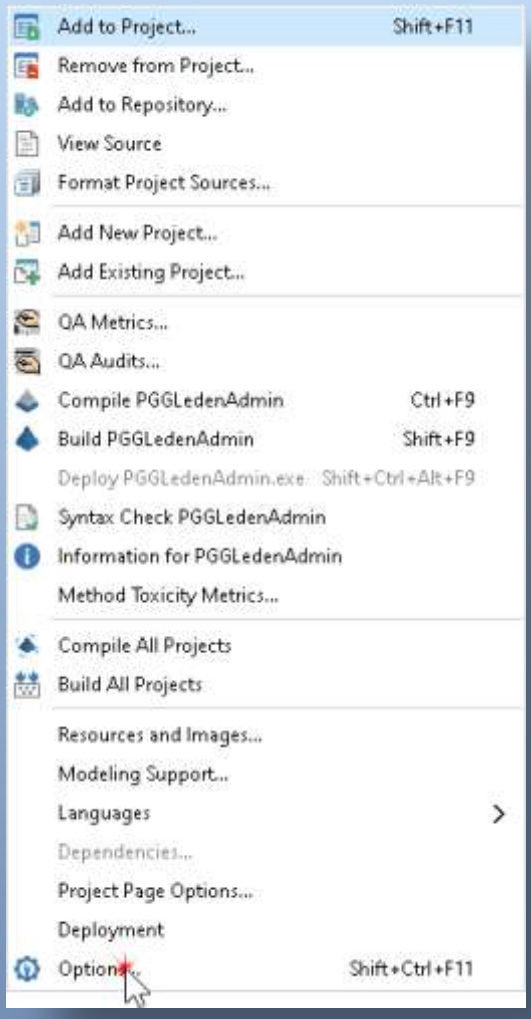



Figure 3:

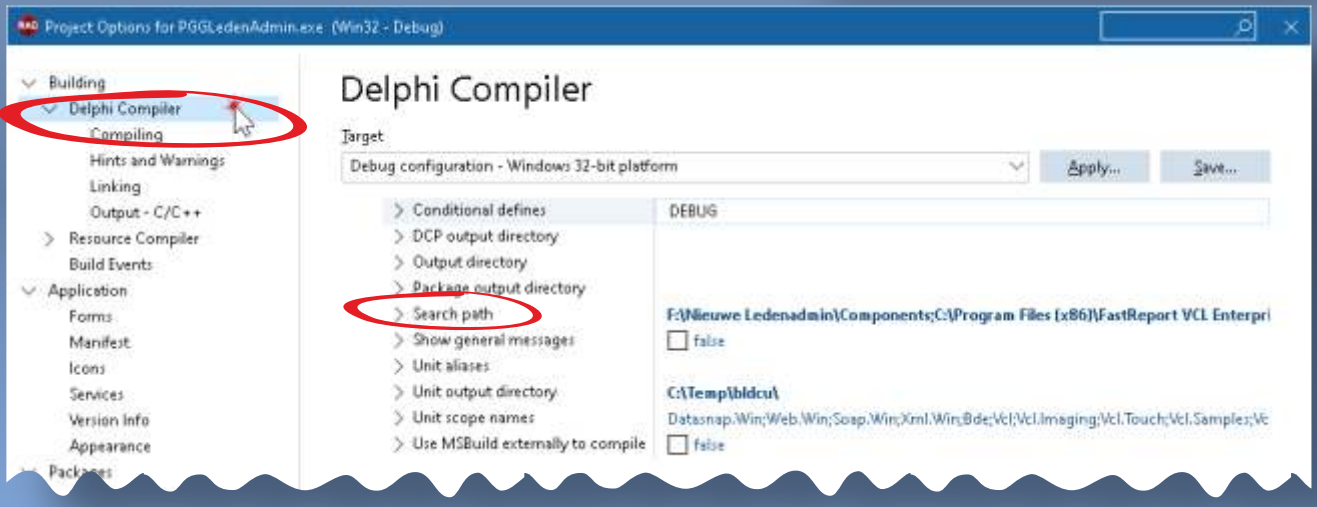



Figure 4:

## COMPONENT SEARCH PATHS

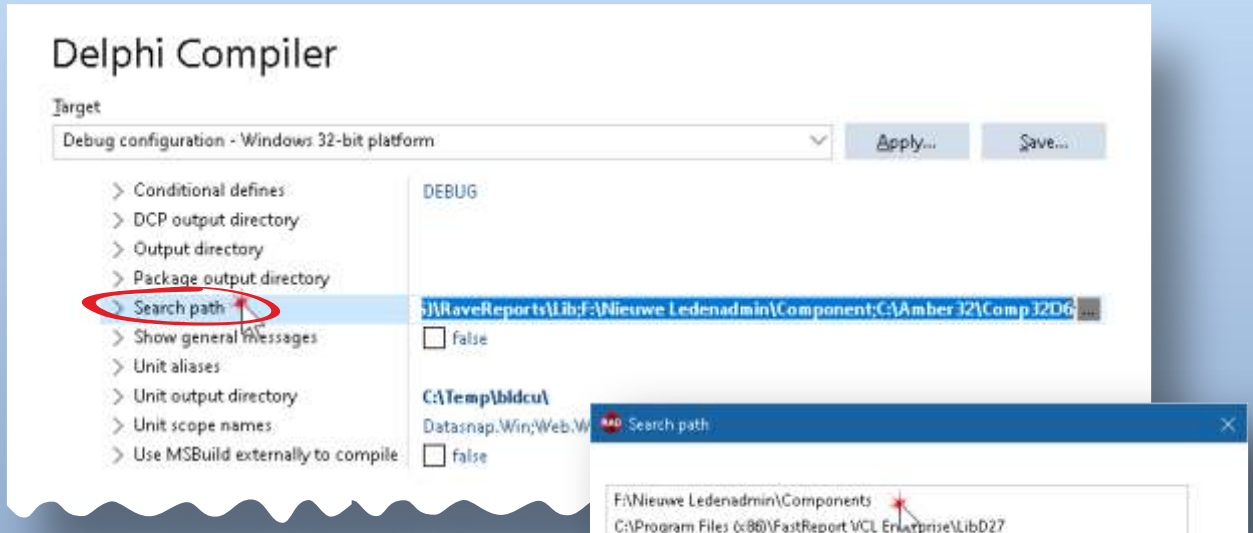



Figure 5:

You can open the search path, but it wont accept changes. So if you really want to alter things you need to click the item open, see below. After that you choose the search path itself a new window opens where you can add, delete and update.

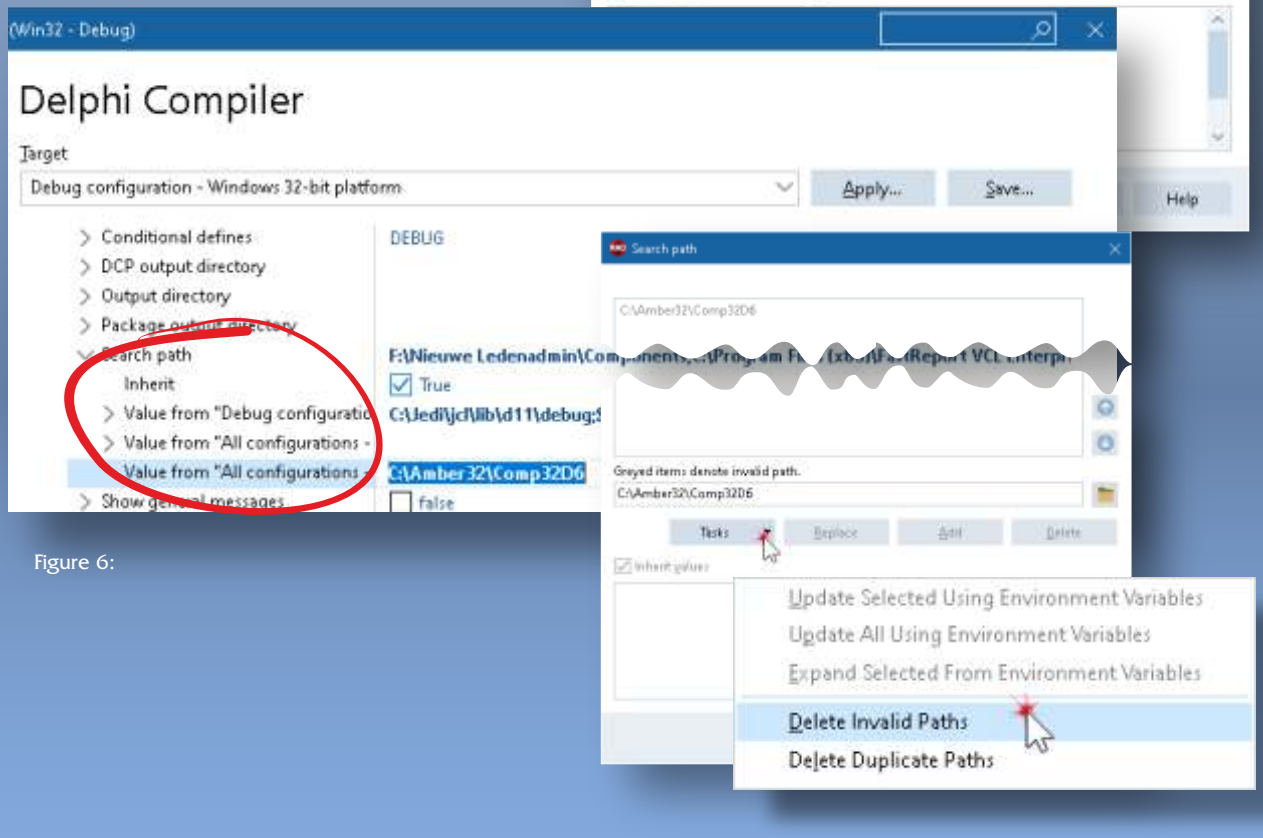



Figure 6:

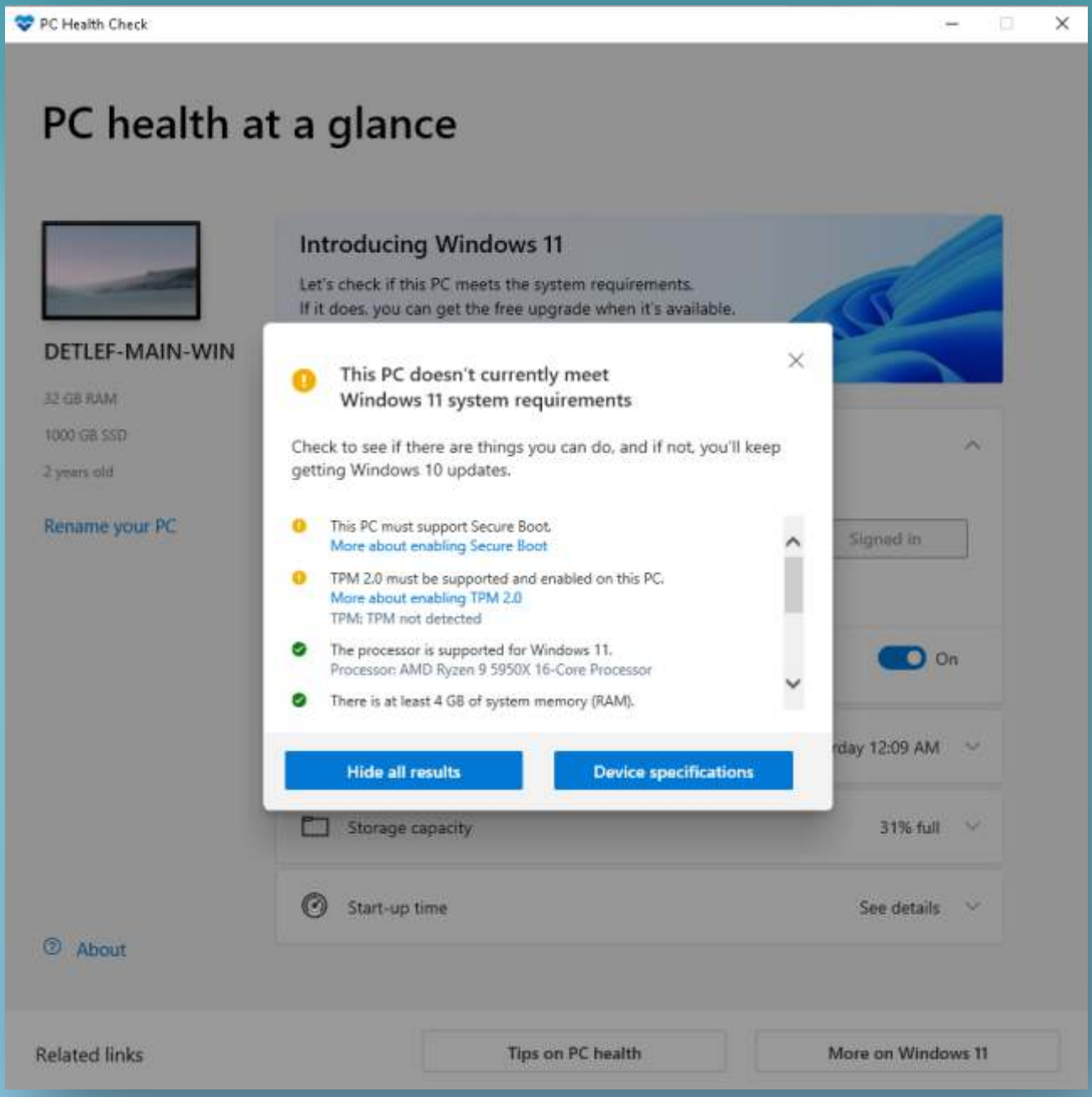



Figure 7:

Because I wanted to test the capability of the combination of **Windows11** and **Delphi11.**
So I started to investigated when the new windows will become available and was advised to download a small program to find whether my computer was capable of using **Windows11.**
`(https://www.microsoft.com/en-us/software-download/windowsinsiderpreviewpchealth)`
The name is **WindowsPCHealthCheckSetup.msi.** Well to be short, it isn't.
*See Figure 7 on page 5.* So I want be able to test this before a final version is presented and that will be about 5 October 2021. I must say if this is true they will press people to buy new machines. That means your clients will probably not start or move to Windows 11.
In the next **Blaise Pascal Magazine** issue Nr. 99/100 I will delve deeper in to that item.

A preview:


Figure 8:

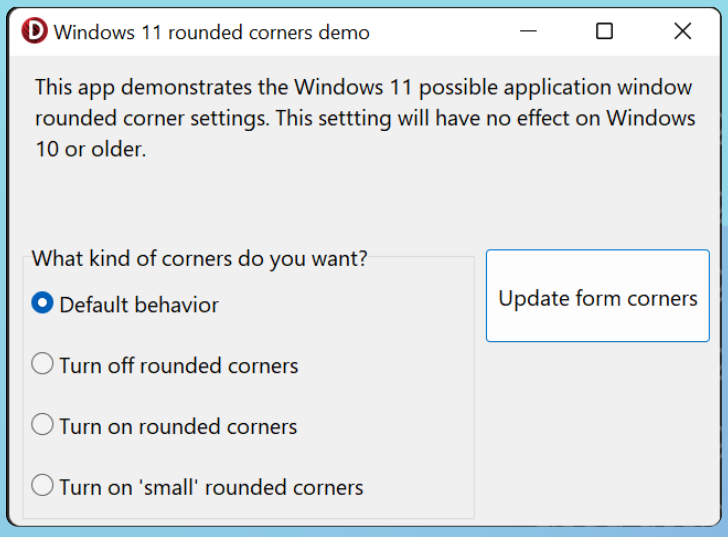



Figure 9:

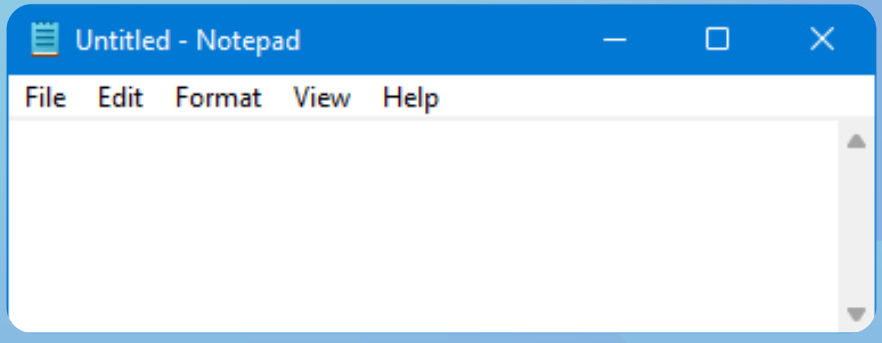



Figure 10:

- **Rich Edit component**
  update removes XP dependencies and introduces new features to `TRichEdit` control
- **VCL Styles add design-time support:**
  prototype stylish UIs even faster by seeing immediately at design-time how your styled forms and controls will look when running
- **Support for CheckBoxes in TreeViews added**, with each node supporting 3 states (Partial, Dimmed, Exclusion) to help customise the UI
- **New `TDBLabeledEdit` component** offering a data-aware version of the `TLabelEdit` for faster prototyping
- **Numerous VCL improvements** including default form size and font, exception dialog copy button, margins on **Memo** and **RichEdit** and many more

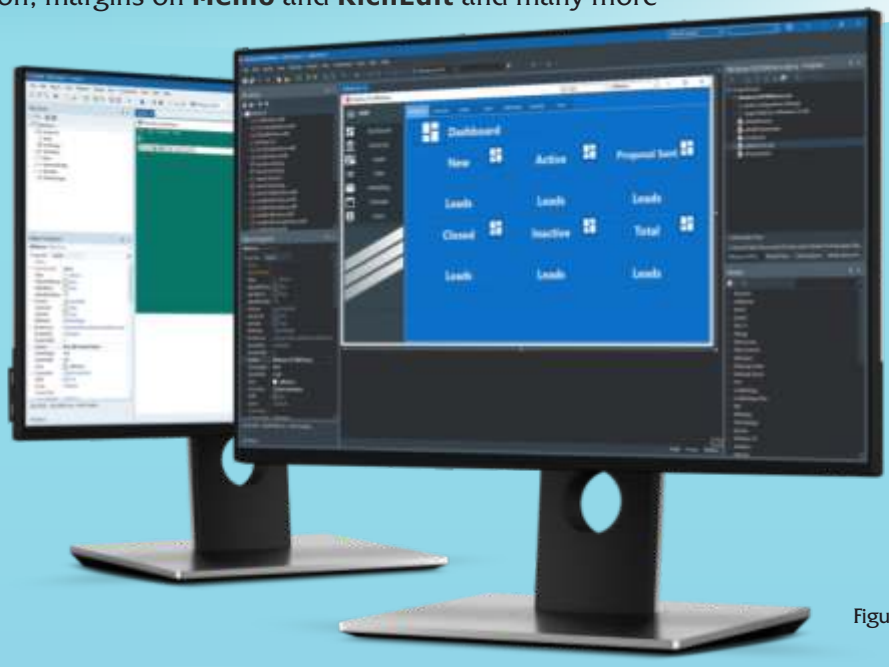



Figure 11:

- **High-DPI support to the IDE**, covering the latest 4k+ monitors, and cleaner and sharper fonts and icons throughout
- **Multi-monitor and multi-window** improvements: design and edit code for the same form at the same time in multiple windows
- **Completely rebuilt Welcome Page** with a native look and feel, and a UI that fits the IDE, and customizable layout and content

- **RTL Quality focus:**
- TZipFile, 64bit improvements for large data structures, Bluetooth LE
- Record Helper for `TDateTime` in `System.DateUtils`
- Record Helper for `TCurrency` in `System.SysUtils` makes working with Currency simpler and easier

- **FireDAC** in the new release offers specific improvements for the **PostgreSQL, Oracle,** and **Firebird** databases
- **HTTP and REST client libraries** have been extended with timeout mechanisms, support for **HTTP/2, TLS 1.3, Base64 URL Encoding**
- **New component `TRESTRequestDataSetAdapter`** simplifies uploading datasets to RAD Server
- **New low-traffic RAD Server Lite** allows unlimited deployment of your multi-tier solutions, alongside with the fully scalable RAD Server engine
- For **DataSnap,** the **REST URL** mapping logic is now fully configurable

**Compile for Android API 30!**
Android API and Libraries updated - API 30, Google Play V3, Android X.
Keep current with the latest requirements for Android as the platform evolves.

This includes the latest billing API (required by Google Play Store).
Enhanced Delphi and C++ RTL for Android, with support for Android API level 30. Support for the new "AndroidX" libraries.
In-app purchase component to help monetize your applications. Android solution Google Play Billing Library Version 4. Enhanced FireMonkey Application Platform for creating native Android ARMv7 applications for Android 11, 10, Pie (9.0), Oreo (8.1)

Figure 12:

## WINDOWS SUBSYSTEM FOR LINUX

**Delphi** has for **Windows11** the enhanced **WSL2** and soon **WSL2** for **Android**
**Delphi 11** support for remote debugging in **WSL2**

WHAT IS THE WINDOWS SUBSYSTEM FOR LINUX?
The **Windows Subsystem for Linux** lets developers run a **GNU/Linux** environment -- including most command-line tools, utilities, and applications -- directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup.
You can:

- **Choose your favorite GNU/Linux** distributions from the **Microsoft Store.**
- **Run common command-line tools** such as grep, sed, awk, or other ELF-64 binaries.
- **Run Bash shell scripts** and GNU/Linux command-line applications including:
  - Tools: vim, emacs, tmux
  - Languages: **NodeJS, Javascript, Python, Ruby, C/C++, C# & F#, Rust, Go**, etc.
  - Services: **SSHD, MySQL, Apache, lighttpd, MongoDB, PostgreSQL**.
- Install additional software using your own **GNU/Linux** distribution package manager.
- Invoke **Windows** applications using a Unix-like command-line shell.
- Invoke **GNU/Linux** applications on **Windows.**


Figure 13:

Improved FMX High-DPI
support for Windows and Desktop with a visibly superior desktop UI

- Support for Microsoft's WebView 2 control (Edge Chromium) in the WebBrowser component
- Support for the latest Android 30 API and latest Billing APIs, and migration to use the AndroidX libraries
- Android support for multiple classes.dex files, simplifying integration of external Android dependencies

DEPLOY ON M-SERIES APPLE SILICON!
Compile for **macOS (M-series Apple Silicon)** and use the new universal package for **AppStore** submission. You can now compile for both existing **Intel** and new **M-series macOS processors (Apple Silicon).** Compiling for the newest processor versions enables the fastest performance across all platforms, and supports universal packaging for the **macOS app** store.
With **RAD Studio 11** it is possible to compile binaries for **macOS ARM.** Since the new **M1** processor is incredibly fast it is more than important to create native apps for it.
That's why **RAD Studio 11** is a must have for every Delphi Developer!

BE READY FOR WINDOWS 11!
Support for Windows 11 provisioning with integrated MSIX generation. WebBrowser component for Windows, with support for both the IE ActiveX and the new Microsoft WebView 2 control (*Chromium-based Edge*).
Enhanced **VCL Form Designer** to visually build native Windows applications, with live snap-to hints and layout guidelines. Enhanced Delphi RTL for 32-bit Windows and 64-bit Windows. (*See the designer next article page 8* →)

**Delphi macOS 64bit ARM compiler** and toolchain
that includes building universal binaries for **Intel/Arm AppStore** submissions.
**Delphi** language support for binary decimals and digits separator.

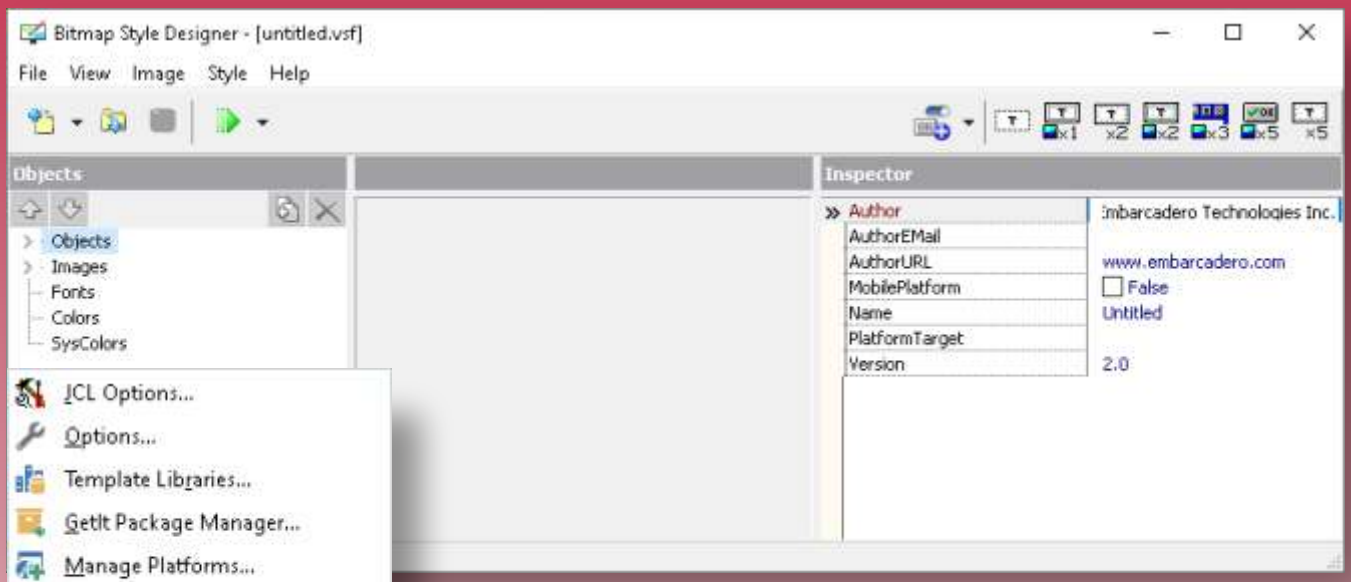



Figure 14:

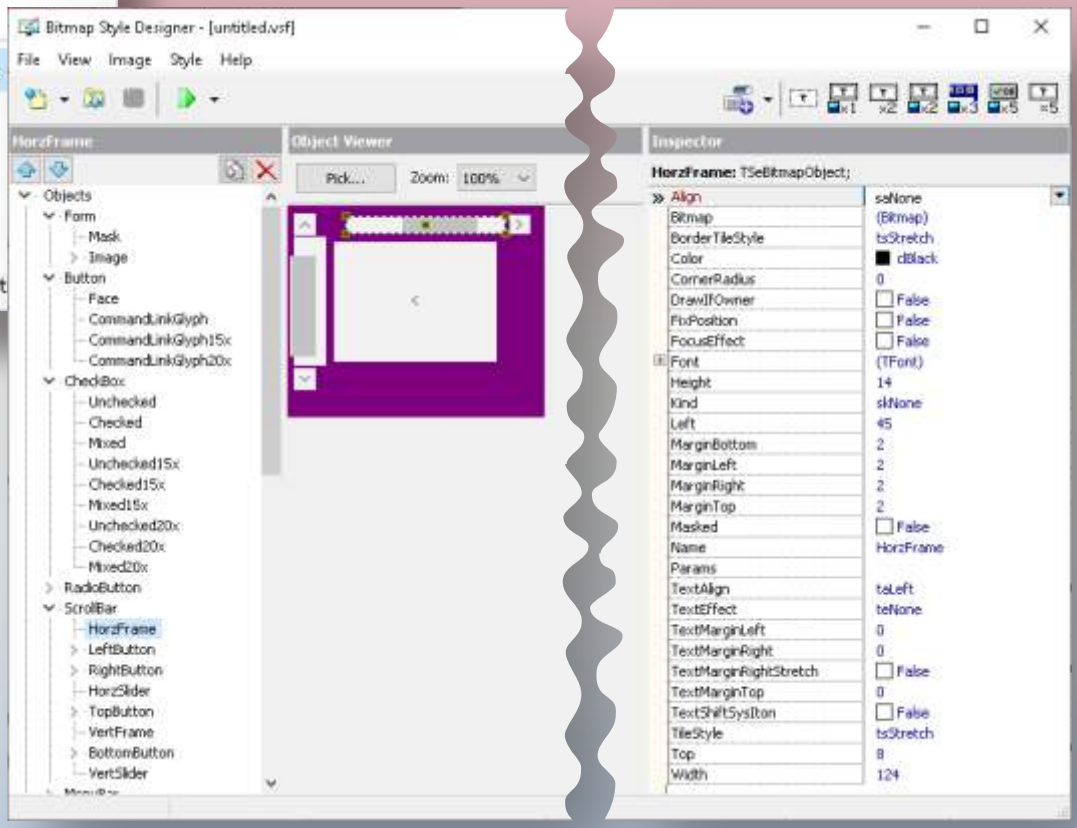



Figure 15:

Figure 16:

First step: go to Tools. The left shown window will pop up.
Then select Bitmap Style Designer The **Bitmap Style Designer** opens. (*See Figure* 16)
On the next page you can see an eventual result and test what you have achieved.

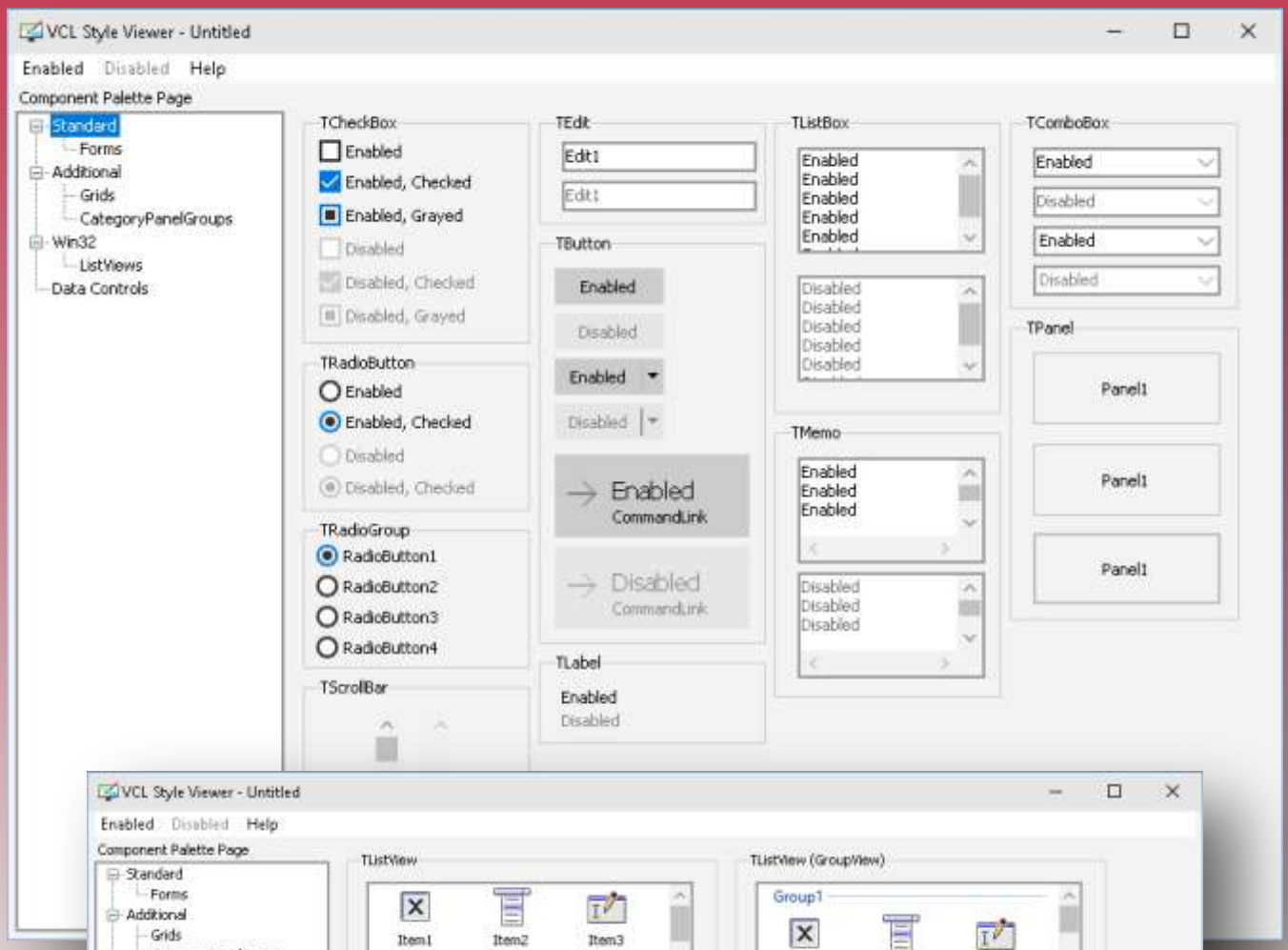



Figure 17:

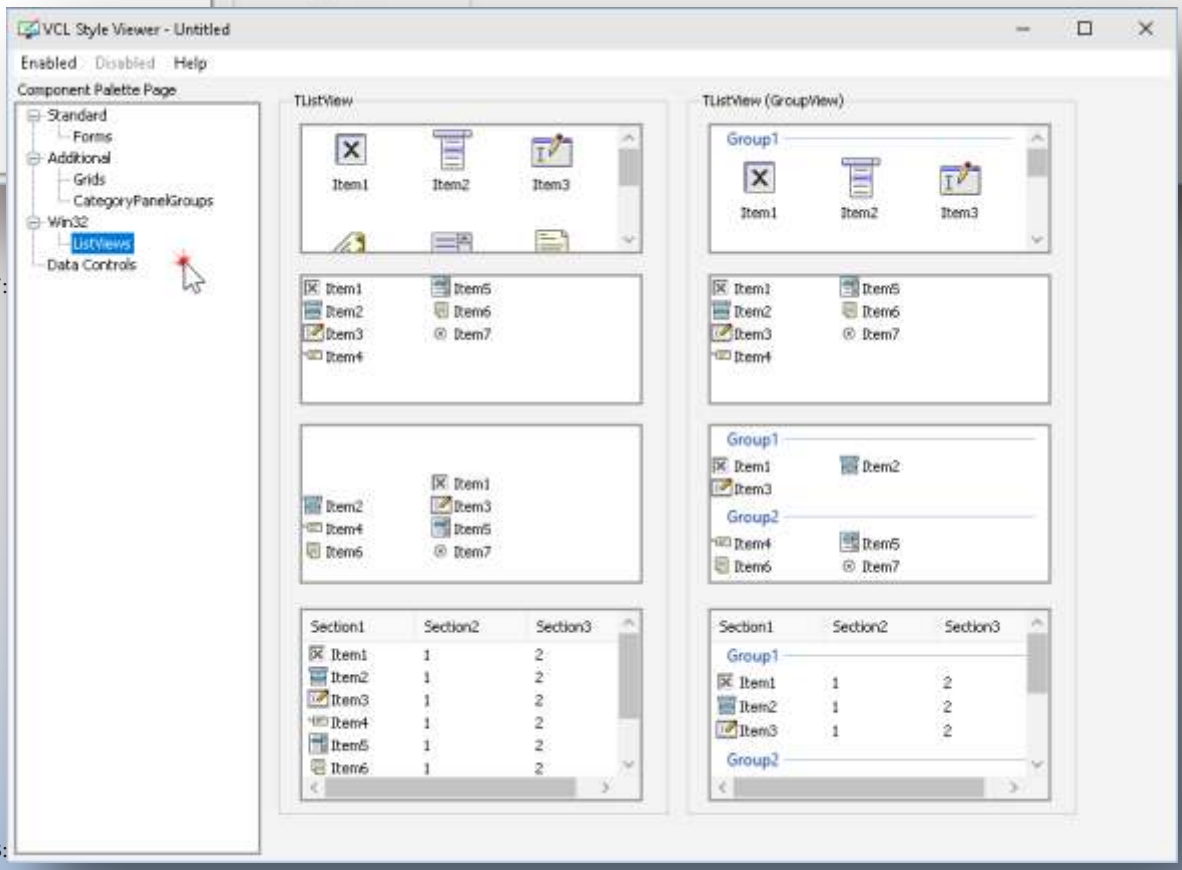



Figure 18:

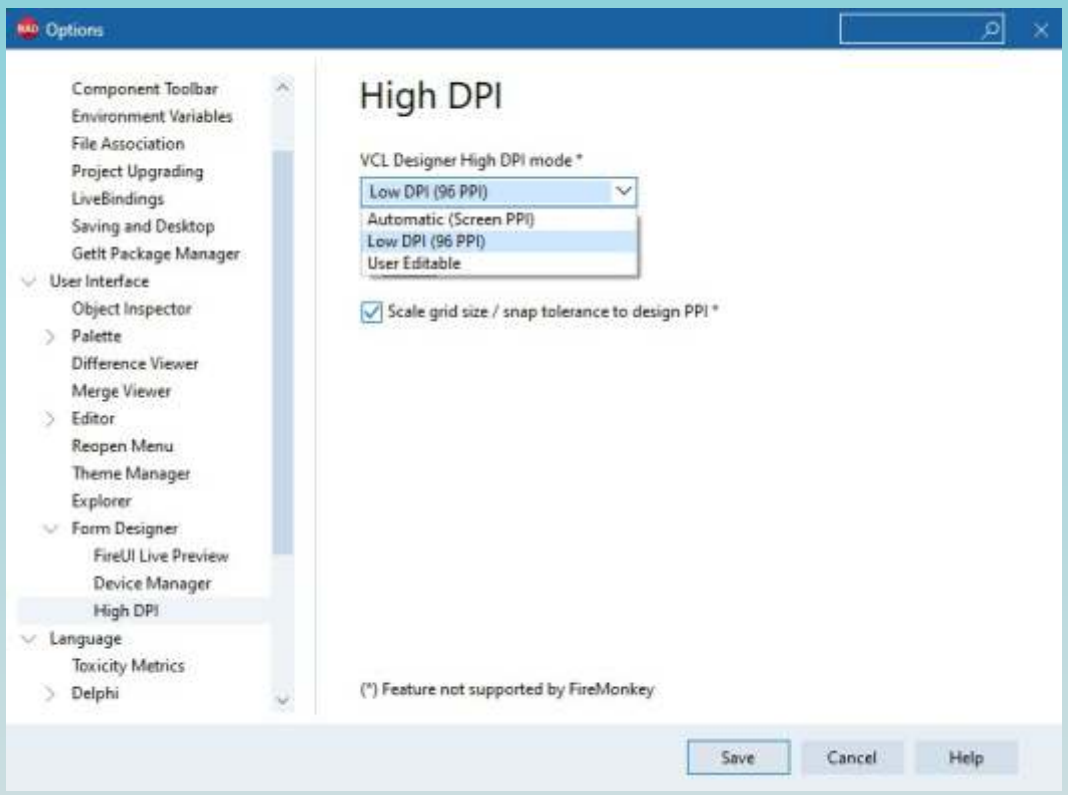



Figure 19:

## VCL DESIGNER DPI MODE

The VCL form designer can handle High DPI scaling in three different ways. Select a value in 'VCL Designer High DPI mode':

- **Automatic (Screen PPI):**
  The designed forms are scaled to the current screen the designer is placed on. It uses the same scale as the other IDE windows on the same screen.

- **Low DPI (96 PPI):**
  This is the default High DPI mode. Forms are scaled to 96DPI, which is 100% scaling for Windows. If your screen has a higher DPI, then the form will be visually smaller than the UI elements of the IDE.

- **User Editable:**
  Choose any DPI to scale forms to.
  When you select the User Editable option, the PixelsPerInch becomes a writable property, and you can enter a PPI value to scale the form to that PPI in the designer.
  If the scaling is set to Automatic or Low DPI, this property becomes IDE-managed and is read-only.

The PixelsPerInch overload property ensures a form is always designed and saved at a specific PPI.

NOTE: Set `Tform.Scaled` to `True` for scaling to take effect in the designer.

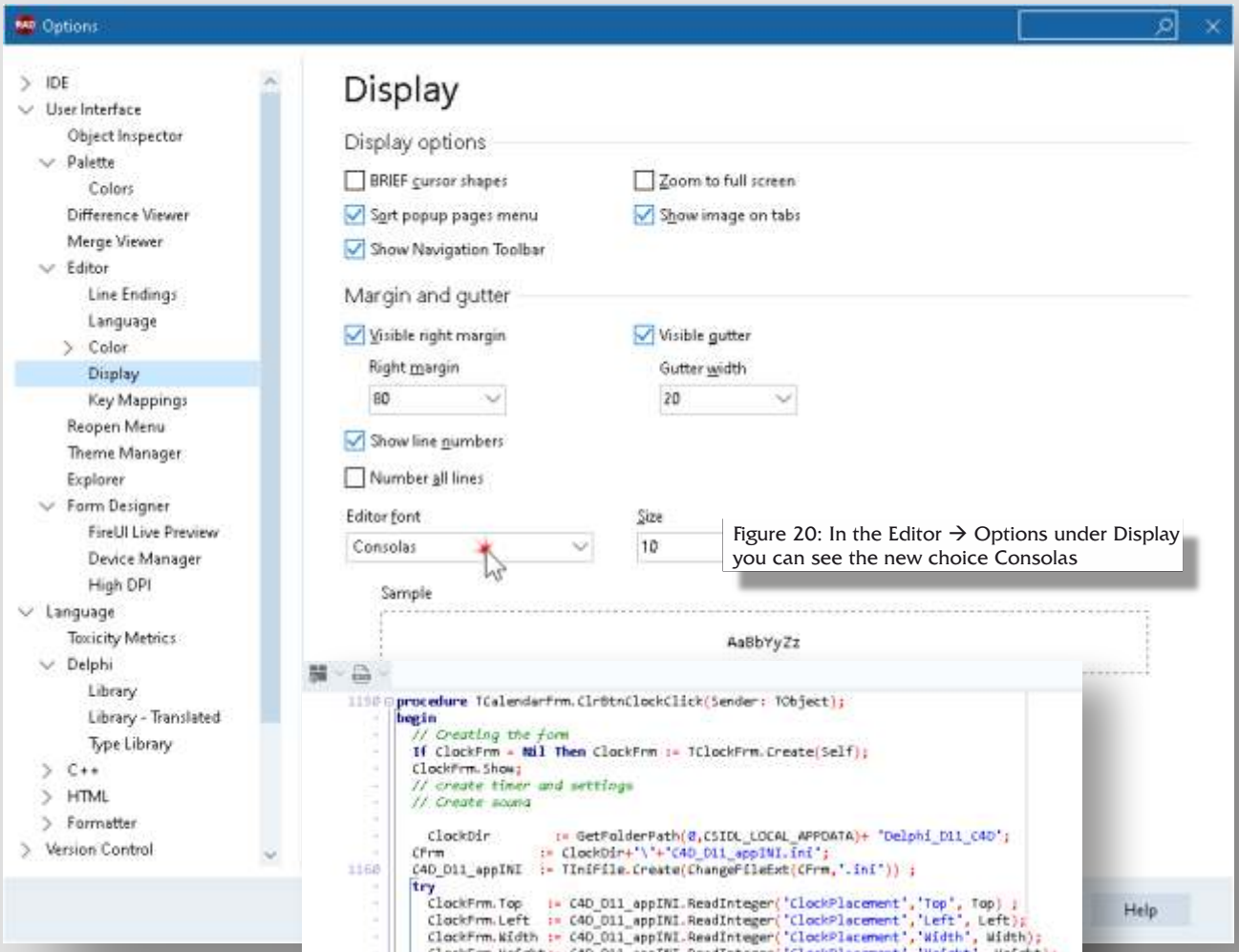



Figure 20: In the Editor → Options under Display you can see the new choice Consolas

```
procedure TCalenderFrm.ClrBtnClockClick(Sender: TObject);
begin
  // Creating the form
  If ClockFrm = Nil Then ClockFrm := TClockFrm.Create(Self);
  ClockFrm.Show;
  // create timer and settings
  // Create ocuna

    ClockDir         := GetFolderPath(0,CSIDL_LOCAL_APPDATA)+ 'Delphi_D11_C4D';
  CFrm             := ClockDir+'\'+'C4D_D11_appINI.ini';
  C4D_D11_appINI   := TIniFile.Create(ChangeFileExt(CFrm,'.ini')) ;
  try
    ClockFrm.Top    := C4D_D11_appINI.ReadInteger('ClockPlacement','Top', Top) ;
    ClockFrm.Left   := C4D_D11_appINI.ReadInteger('ClockPlacement','Left', Left);
    ClockFrm.Width  := C4D_D11_appINI.ReadInteger('ClockPlacement','Width', Width);
    ClockFrm.Height := C4D_D11_appINI.ReadInteger('ClockPlacement','Height', Height);
  finally
    C4D_D11_appINI.Free;
  end;
  visible:= False;
end;
```

Figure 21:

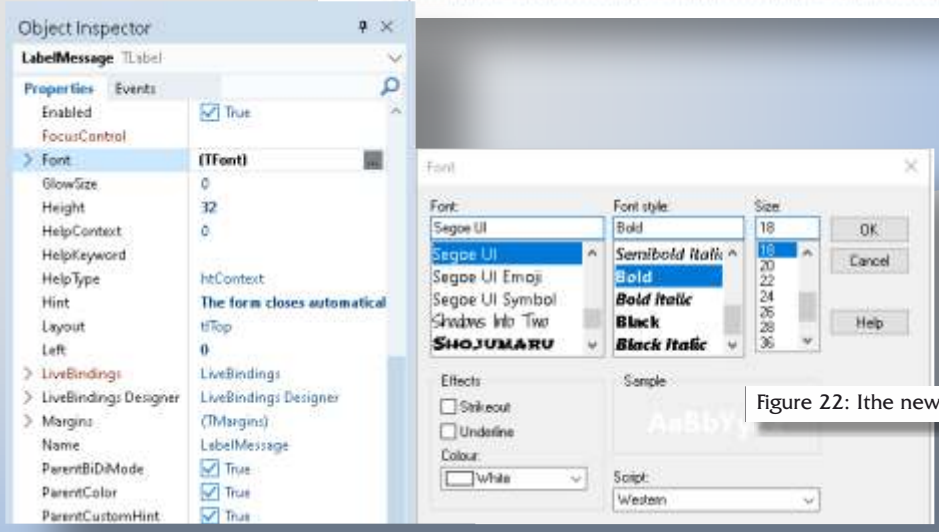



Figure 22: Ithe new standard font: Segou UI

**This setting applies to all forms that the IDE opens.**
Forms and controls are scaled at design time using the same `perMonitorv2 high DPI https://docwiki.embarcadero.com/RADStudio/Alexandria/en/IDE_Command_Line_Switches_and_Options#General_options_for_BDS.EXE` support that is used when scaling at runtime.

The design DPI is saved in the form DFM and affects the pixel coordinates and sizes of controls (such as Left and Height.) This means that opening a form changes the values that will be saved to the DFM file. If your team uses multiple different scaling settings among team members, you can set the User Editable option on each computer to have a consistent common scaling for all forms among all team members.
Attention: When designing forms that will run at High DPI, do not turn ParentFont to True for a form.

**Scale grid size / snap tolerance to design PPI**
You can change the visual grid on a form by selecting the Scale grid size / snap tolerance to design PPI option.
By default, this draws a dot every 8x8 pixels, though this is configurable. When this checkbox is checked, the value (e.g. 8x8) is scaled the same as the form.
**High DPI for the FMX Form Designer**
The FireMonkey designer scales a form the same way an FMX form does at runtime. Coordinates remain the same, it is only a visual scaling. That means that a button placed at (20, 20) will be at (20, 20) no matter the high DPI scaling of the monitor the FMX designer is on.

## HTTP/2

Delphi has update to **HTTP/2** (originally named **HTTP/2.0**) is a major revision of the `HTTP` network protocol used by the `World Wide Web`.
It was derived from the earlier experimental **SPDY** protocol, originally developed by **Google.**
**HTTP/2** was developed by the HTTP Working Group (also called httpbis, where "bis" means "twice") for the Internet **Engineering Task Force (IETF).**
**HTTP/2** is the first new version of **HTTP** since **HTTP/1.1**

The standardization effort was supported by **Chrome, Opera, Firefox, Internet Explorer 11, Safari, Amazon Silk**, and **Edge** browsers.
Most major browser had added **HTTP/2** support by the end of 2015.
About 97% of web browsers used have the capability.
Its proposed successor is HTTP/3, a major revision that builds on the concepts established by HTTP/2.
The working group charter mentions several goals and issues of concern:

## HTTP/2 CONTINUATION

- Create a negotiation mechanism that allows clients and servers to elect to use HTTP/1.1, 2.0, or potentially other non-HTTP protocols.
- Maintain high-level compatibility with HTTP/1.1 (*for example with methods, status codes,URIs, and most header fields*).
- Decrease latency to improve page load speed in web browsers by considering: data compression of HTTP headers
- HTTP/2 Server Push
- pipelining of requests
- fixing the head-of-line blocking problem in HTTP 1.x
- multiplexing multiple requests over a single TCP connection
- Support common existing use cases of HTTP, such as desktop web browsers, mobile web browsers, web APIs, web servers at various scales, proxy servers, reverse proxy servers, firewalls, and content delivery networks.

Differences from HTTP
The proposed changes do not require any changes to how existing web applications work, but new applications can take advantage of new features for increased speed. HTTP/2 leaves all of HTTP/1.1's high-level semantics, such as methods, status codes, header fields, and URIs, the same. What is new is how the data is framed and transported between the client and the server.

Websites that are efficient minimize the number of requests required to render an entire page by minifying (*reducing the amount of code and packing smaller pieces of code into bundles, without reducing its ability to function*) resources such as images and scripts. However, minification is not necessarily convenient nor efficient and may still require separate HTTP connections to get the page and the minified resources. HTTP/2 allows the server to "push" content, that is, to respond with data for more queries than the client requested. This allows the server to supply data it knows a web browser will need to render a web page, without waiting for the browser to examine the first response, and without the overhead of an additional request cycle.

Additional performance improvements in the first draft of HTTP/2 (which was a copy of SPDY) come from multiplexing of requests and responses to avoid some of the head-of-line blocking problem in HTTP 1 (even when HTTP pipelining is used), header compression, and prioritization of requests. However, as HTTP/2 runs on top of a single TCP connection there is still potential for head-of-line blocking to occur if TCP packets are lost or delayed in transmission. HTTP/2 no longer supports HTTP/1.1's chunked transfer encoding mechanism, as it provides its own, more efficient, mechanisms for data streaming.

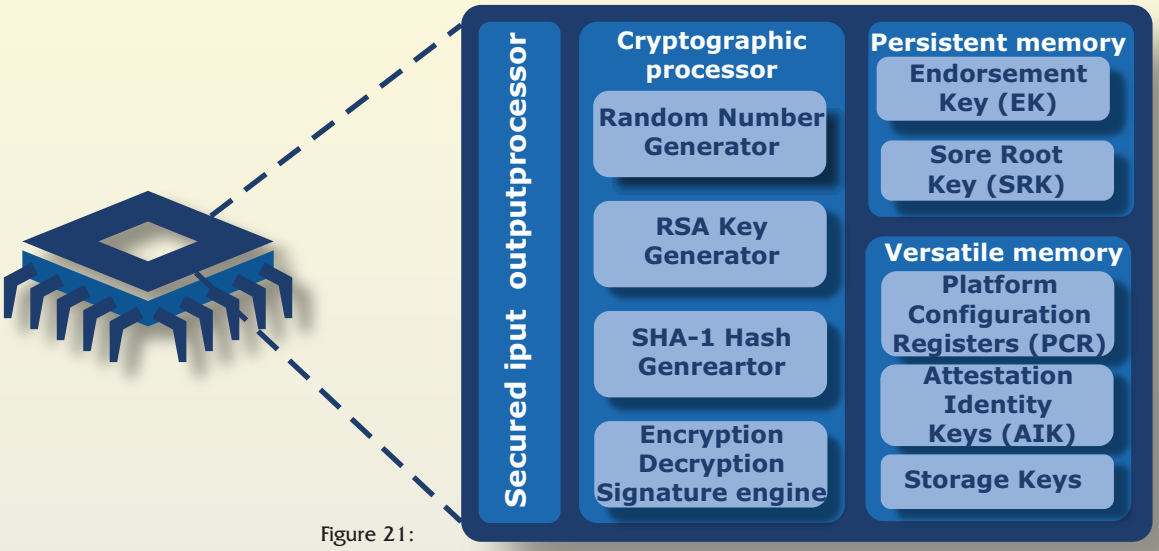



Figure 21:

WINDOWS11 REQUIRES A NEW SAFER HARDWARE (INCLUDING TPM).
**Trusted Platform Module (TPM, also known as ISO/IEC 11889)** is an international standard for a **secure cryptoprocessor**, a **dedicated microcontroller** designed to **secure hardware through integrated cryptographic keys.** The term can also refer to a chip conforming to the standard.

**Trusted Platform Module (TPM)** technology is designed to provide hardware-based, security-related functions. A TPM chip is a secure crypto-processor that is designed to carry out cryptographic operations. The chip includes multiple physical security mechanisms to make it tamper-resistant, and malicious software is unable to tamper with the security functions of the TPM.

SOME OF THE KEY ADVANTAGES OF USING TPM TECHNOLOGY ARE THAT YOU CAN:

- **Generate, store, and limit** the use of cryptographic keys.
- **Use TPM technology for platform device authentication** by using the TPM's unique RSA key, which is burned into itself.
- **Help ensure platform integrity** by taking and storing security measurements.

The most common TPM functions are used for system integrity measurements and for key creation and use. During the boot process of a system, the **boot code that is loaded** (including firmware and the operating system components) can be measured and recorded in the **TPM**. The integrity measurements can be used as evidence for how a system started and to make sure that a **TPM-based key was used only when the correct software was used to boot the system.** TPM-based keys can be configured in a variety of ways.

**One option is to make a TPM-based key unavailable outside the TPM. This is good to mitigate phishing attacks because it prevents the key from being copied and used without the TPM.**

TPM-based keys can also be configured to require an authorization value to use them. If too many incorrect authorization guesses occur, the TPM will activate its dictionary attack logic and prevent further authorization value guesses. Different versions of the TPM are defined in specifications by the Trusted Computing Group (TCG). For more information, consult the `TCG Web` site:
`https://trustedcomputinggroup.org/work-groups/trusted-platform-module/`

# THE INTRODUCTION OF THE NEXT VERSION LSP (LANGUAGE SERVER PROTOCOL)


The code completion **LSP (Language Server Protocol)** server now has a plugin for **VSCode (Visual Studio Code)** and is more in line with the standard.

Note, classic code completion has now been removed.

1. Use of Visual Studio Code to edit Delphi source with full code completion
2. LSP awareness of Include files
3. Auto restart of LSP server
4. Auto code completion with Tab key
5. Class helper support
6. Array suggestions when assigning arrays

Langserver.org (`https://langserver.org`)
A community-driven source of knowledge for Language Server Protocol implementations (`https://microsoft.github.io/language-server-protocol/`)

Langserver.org is a community-driven site, maintained by Sourcegraph, to track development progress of **LSP**-compatible language servers and clients.

What is **LSP**?
The **Language Serve**r protocol is used between a tool (the client) and a language smartness provider (the server) to integrate features like auto complete, go to definition, find all references and alike in the tool – official **Language Server Protocol** specification
The **LSP** was created by **Microsoft** to define a common language for programming language analyzers to speak.

Today, several companies have come together to support its growth, including **Codenvy, Red Hat**, and **Sourcegraph,** the protocol will be supported by a rapidly growing list of editor and language communities.
Look at the details on and links to current client and server implementations.

WHY LSP?
**LSP** creates the opportunity to reduce the m-times-n complexity problem of providing a high level of support for any programming language in any editor, IDE, or client endpoint to a simpler m-plus-n problem.

For example, instead of the traditional practice of building a Python plugin for VSCode, a Python plugin for Sublime Text, a Python plugin for Vim, a Python plugin for Sourcegraph, and so on, for every language, **LSP allows language communities to concentrate their efforts on a single, high performing language server that can provide code completion**, hover tooltips, jump-to-definition, find-references, and more, while editor and client communities can concentrate on building a single, high performing, intuitive and idiomatic extension that can communicate with any language server to instantly provide deep language support.



# kbmMW Community Edition v. 5.16.00 released for Delphi 10.4.2 Sydney!!

**THIS IS A MAJOR NEW RELEASE WITH THE INTRODUCTION OF OUR NEW CONTEXT SENSITIVE I18N INTERNATIONALIZATION FRAMEWORK.**

Components4Developers has decided to let the world have the opportunity to play with the hundreds of thousands of lines of code, containing thousands of functions, procedures, methods, classes and types, covering the vast number of features that makes kbmMW the leader of n-tier development environments.


We have now released our **second Community Edition** which contains almost all features from kbmMW Enterprise Edition, but limited in some areas, partly due to technical limitations, partly due to a few artificial limitations.
It further contains kbmMemTable Community Edition which is a direct match of kbmMemTable Standard Edition.

Community Edition is FREE to use as long as the applications it is used in provides a direct or indirect revenue not reaching US$5000 and the company distributing/producing the applications have a yearly turnover of less than US$50000.

**Read the `license.txt` file for all details.**

Community Edition can only be used for producing 32 bit Windows applications, but you have access to SmartServices, REST, WIB, kbmMemTable, SQL, ORM, SmartEvent, SmartBinding, Scheduler, Logging, Configuration, Ciphers, transports, XML, JSON, YAML, BSON, MessagePack, i18n and the countless of other features you have read about here on the blog!

Community Edition can thus also be used as a 60 day trial for kbmMW Enterprise Edition.
Community Edition does not contain source code, and will only work together with the version of Delphi it was released for.

Current release match latest Delphi Sydney 10.4.2, including Community Edition.

If you want go further, then purchase kbmMW Enterprise Edition which really is the best value for money and while being the oldest existing, mostly backwards compatible, n-tier framework for Delphi, it is also the only one bringing you to the bleeding edge of current technology.

Sign up at **`https://portal.components4developers.com and download immediately.`**

—-
kbmMW is the premiere **n-tier product** for Delphi, C++Builder and FPC on Win32, Win64, Linux, Java, PHP, Android, IOS, .Net, embedded devices, websites, mainframes and more. Please visit
**`http://www.components4developers.com for more information about kbmMW.`**

**Components4Developers is a company established in 1999 with the purpose of providing high quality development tools for developers and enterprises. The primary focus is on SOA, EAI and systems integration via our flagship product kbmMW.**

**kbmMW is currently used as the backbone in hundreds of central systems, in hospitals, courts, private, industries, offshore industry, finance, telecom, governments, schools, laboratories, rentals, culture institutions, FDA approved medical devices, military and more.**

COMPONENTS 4 DEVELOPERS

DX


# KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.18.00 RELEASED!

- **RAD Studio XE5 to 10.4.2 Sydney supported**
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support
- Native high performance 100% developer defined application server
- Full support for centralized and distributed load balancing and failover
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multithread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronouncable password generators.
- High performance LZ4 and Jpeg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, JSON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

**kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.**

- **Easily supports large datasets with millions of records**
- **Easy data streaming support**
- **Optional to use native SQL engine**
- **Supports nested transactions and undo**
- **Native and fast build in M/D, aggregation/grouping, range selection features**
- **Advanced indexing features for extreme performance**

- New I18N context sensitive internationalization framework to make your applications multilingual.
- New ORM LINQ support for Delete and Update.
- Comments support in YAML.
- New StreamSec TLS v4 support (by StreamSec)
- Many other feature improvements and fixes.

**Please visit**
**http://www.components4developers.com**
**for more information about kbmMW**

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.


COMPONENTS 4 DEVELOPERS

**EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER**